НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»
Науковий керівник кафедри
_____ І.А. Дичка
«___»_____2019 р.

Дипломний проект

на здобуття ступеня бакалавра
з напряму підготовки 6.050103 «Програмна інженерія»
на тему: «Програмне забезпечення для аналізу ефективності Інтернет-магазинів на основі статистичної обробки даних»

Виконав:
Студент IV курсу, групи КП-52
Єременко Владислав Євгенійович _____

Керівник:
Ст. викладач кафедри ПЗКС,
Сущук-Слюсаренко В.І. _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н.,
Онай М.В. _____

Рецензент:
Доцент кафедри СПСКС, к.т.н.,
Сапсай Т.Г. _____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2019 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки (програма професійного спрямування) – 6.050103 «Програмна інженерія»

ЗАТВЕРДЖУЮ

Науковий керівник кафедри

_____ І.А. Дичка

«___»_____ 2018 р.

ЗАВДАННЯ

на дипломний проект студенту

Єременку Владиславу Євгенійовичу

1. Тема роботи «Програмне забезпечення для аналізу ефективності інтернет-магазинів на основі статистичної обробки даних», керівник проекту Вікторія Ігорівна Сушук-Слюсаренко, старший викладач, затверджені наказом по університету від «22»травня 2019 р. №1331-С.

2. Термін подання студентом проекту «__» червня 2019 р.

3. Вихідні дані для дипломного проектування: див. Технічне завдання.

4. Зміст проекту:
   - аналіз методу флуктуацій;
   - вибір методу побудови моделі;
   - побудова статистичної моделі для прогнозування кількості продажів;
   - перевірка моделі для прогнозування кількості продажів.
   - розробка програмного забезпечення з графічним інтерфейсом, яке буде здатне моделювати результати зміни параметрів відповідної моделі

5 Перелік обов'язкового ілюстративного матеріалу:
   - Використання статистичних алгоритмів в залежності від режиму роботи (креслення);
   - Обробка даних та проведення оцінки (креслення);
   - Оцінка роботи статистичних методів (плакат);
   - Ілюстрація роботи статистичних методів (плакат);

6. Консультанти розділів роботи

Розділ		Підпис, дата

	Прізвище, ініціали та посада консультанта	завдання видав	завдання прийняв
	Онай М.В., доцент кафедри ПЗКС, к.т.н.		

7. Дата видачі завдання «31» жовтня 2018 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1.	Вивчення літератури за тематикою проекту та збір даних	16.11.2018	
2.	Розроблення та узгодження технічного завдання	09.12.2018	
3.	Підготовка матеріалів першого розділу дипломного проекту	30.12.2018	
4.	Розробка математичної моделі для прогнозування тривалості життя населення	16.01.2019	
5.	Підготовка матеріалів другого розділу дипломного проекту	10.02.2018	
6.	Підготовка матеріалів третього розділу дипломної роботи	15.03.2019	
7.	Підготовка матеріалів третього розділу дипломного проекту	13.04.2019	
	Підготовка матеріалів четвертого розділу дипломного проекту	11.04.2019	
8.	Підготовка графічної частини дипломного проекту	19.05.2019	
9.	Оформлення документації дипломного проекту	26.05.2019	

Студент В.Є Єременко

Керівник роботи В.І. Сушук-Слюсаренко

АНОТАЦІЯ

Даний проект присвячений дослідженню використання статистичних методів для створення програмного забезпечення, яке б оцінювало кількість продажів при зміні ціни та рекламних платформ.

У проекті було розроблене спеціалізоване програмне забезпечення та виконаний аналіз ефективності використання маржинальних флуктуацій (періодичних випадкових змін націнки на товари) як доцільного джерела даних, необхідних для утворення статистичної моделі, яка здатна коректно оцінити ефект зміни націнки на товари. Результатом дослідження є рекомендація не вживати цей метод у даному вигляді, тим не менш, є ймовірність що його можливо модифікувати чи перевірити на замірах довших за один рік. Проте під час запису даних економічні наслідки від зміни націнки невідомі, тому на великих при довгих замірах доцільність вживання методу сумнівна.

Досліджена можливість використання статистичних методів для оцінки динаміки продажів в залежності від набору рекламних платформ якими інтернет-магазин користується у даний момент часу. У даній моделі була використана лінійна регресія, тому кожний рекламодавець має свій коефіцієнт залежності продажів від витрат на рекламу.

Під час виконання дипломного проекту було розроблене програмне забезпечення з користувацьким інтерфейсом. Функціонал програми дозволяє корегувати параметри інтернет магазину: націнку на товари, кількість витрат на рекламні платформи та стан користування ними.

Результатом зміни параметрів є відображення оцінки кількості продажів.

ABSTRACT

Purpose of this project is to research statistical methods in terms of producing software, which would be able to predict e-shop sales based on price and advertisement providers change.

In this project specialized software was developed and was conducted analysis of marginal fluctuations (repetitive random changes in goods price) as appropriate source of data to create a statistical model, which could correctly predict impact of price change on sales. Result of this research tells that usage of this method is not recommended but it can be modified or tested on scale of more than one year. Although it is hard to predict financial causes of random price change, so big time scales can be inappropriate to use.

There was conducted research of statistical methods usage for sales dynamics prediction based on chosen advertisers. This model is based on linear regression so every advertiser has its own sales/cost coefficient.

In this work, developed software has an UI. Program functionality gives user ability to see sales prediction in real time base on changed parameters: price, advertisement spends and advertising platforms turned on.

ДП.045440-01-90 Програмне забезпечення для аналізу ефективності інтернет-магазинів на основі статистичної обробки даних. Відомість проекту

Позначення	Найменування	Кіл-ть	Примітка
	Документація проекту		
ДП.045440-02-91	Програмне забезпечення	4	
	для аналізу ефективності		
	інтернет-магазинів на		
	основі статистичної		
	обробки даних.		
	Технічне завдання		
ДП.045440-03-81	Програмне забезпечення	57	
	для аналізу ефективності		
	інтернет-магазинів на		
	основі статистичної		
	обробки даних.		
	Пояснювальна записка		
ДП.045440-04-51	Програмне забезпечення	4	
	для аналізу ефективності		
	інтернет-магазинів на		
	основі статистичної		
	обробки даних.		
	Програма та методика		
	тестування		
ДП.045440-05-34	Програмне забезпечення	4	
	для аналізу ефективності		
	інтернет-магазинів на		
	основі статистичної		
	обробки даних.		
	Керівництво користувача		

Позначення	Найменування	Кіл-ть	Примітка
ДП.045440-06-99	Програмне забезпечення	1	
	для аналізу ефективності		
	інтернет-магазинів на		
	основі статистичної		
	обробки даних.		
	Використання		
	статистичних методів в		
	залежності від режиму		
	роботи. Схема алгоритму		
ДП.045440-07-99	Програмне забезпечення	1	
	для аналізу ефективності		
	інтернет-магазинів на		
	основі статистичної		
	обробки даних.		
	Обробка даних		
	та проведення оцінки.		
	Схема алгоритму.		
ДП.045440-08-98	Програмне забезпечення	1	
	для аналізу ефективності		
	інтернет-магазинів на		
	основі статистичної		
	обробки даних.		
	Компакт-диск		

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2018 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ АНАЛІЗУ ЕФЕКТИВНОСТІ ІНТЕРНЕТ-МАГАЗИНІВ НА ОСНОВІ СТАТИСТИЧНОЇ ОБРОБКИ ДАНИХ

Технічне завдання

ДП.045440-02-91

"ПОГОДЖЕНО"

Керівник проекту:

_____ В.І. Сущук-Слюсаренко

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____Єременко В.Є.

2018

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Програмне забезпечення для аналізу ефективності інтернет-магазинів на основі статистичної обробки даних.

Галузь застосування: інформаційні технології.

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на написання дипломної роботи, затверджене кафедрою програмного забезпечення комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім. Ігоря Сікорського).

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Дослідження та розробка направлені на створення програмного забезпечення здатного оцінювати ефективність роботи інтернет-магазинів. Проект націлений першим чином на малі інтернет-підприємства.

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Програмне забезпечення повинні мати наступний функціонал:

1. Редагування рівня націнки;
2. Редагування стану користування рекламними платформами;
3. Зміна витрат на рекламні платформи;
4. Оцінка кількості продажів при зміні параметрів;

5. ВИМОГИ ДО ПРОЕКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проекту повинна бути розроблена наступна документація:

1) пояснювальна записка;

2) програма та методика тестування;

3) керівництво користувача;

4) креслення:

- «Блок-схема використання статистичних методів в залежності від режиму роботи»;
- «Обробка даних та проведення оцінки».

6. ЕТАПИ ПРОЕКТУВАННЯ

Вивчення літератури за тематикою проекту 16.11.2018

Розроблення та узгодження технічного завдання..... 09.12.2018

Підготовка матеріалів першого розділу дипломного проекту 30.12.2018

Розроблення алгоритму прогнозування курсу криптовалют .. 16.01.2019

Підготовка матеріалів другого розділу дипломного проекту . 10.02.2019

Підготовка третього розділу дипломного проекту..... 10.03.2019

Підготовка четвертого розділу дипломного проекту..... 11.04.2019

Підготовка графічної частини дипломного проекту 19.05.2019

Оформлення документації дипломного проекту 26.05.2019

7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ І.А. Дичка

"___" _____ 2019 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ АНАЛІЗУ ЕФЕКТИВНОСТІ ІНТЕРНЕТ-МАГАЗИНІВ НА ОСНОВІ СТАТИСТИЧНОЇ ОБРОБКИ ДАНИХ

Пояснювальна записка

ДП.045440-03-81

"ПОГОДЖЕНО"

Керівник проекту:

_____ В.І. Сущук-Слюсаренко

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____Єременко В.Є.

2019

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

CSV (comma separated values) – формат зберігання даних, де параметри об’єкту розділені комами, а самі об’єкти розривами строки.

Open source – відкрите програмне забезпечення, тобто з загальнодоступним вихідним кодом. Вихідний код таких програмних продуктів загальнодоступний для перегляду, вивчення, зміни та використання.

Прайс-агрегатор (price-aggregator / price-comparison engine / сайт-агрегатор цін) – програмне забезпечення, яке аналізує ціни онлайн-магазинів і виставляє продукти с оптимальними цінами.

Python – універсальна високорівнева процедурна мова програмування с елементами ООП. Найпопулярніша у використанні машинного навчання з розповсюджених універсальних мов програмування.

Статистичні методи – математичні алгоритми, направлені на використання об’ємних масивів даних з ціллю отримання деякої інформації, якість якої збільшується з розміром масиву даних, який поступає на вхід.

Pandas – модуль (бібліотека) мови програмування Python, він надає доступ до програмного інтерфейсу, функціоналом якого є зчитування, маніпуляція та зберігання даних у вигляді схожому на функціонал звичайних реляційних баз даних.

Qt – програмний інструментарій, який надає змогу створення універсального програмного забезпечення з графічним інтерфейсом.

PyQt5 – модуль мови програмування Python, він надає інструментарій, який транслює файли інтерфейсу Qt у файли програмного коду, таким чином

Пайплайн – послідовність застосування декількох етапів обробки даних чи виконання функцій.

Sklearn – модуль мови програмування Python, якій містить в собі програмні реалізації багатьох статистичних методів, а також трансформації та обробки даних для їх використання.

ВСТУП

За декілька століть комерційна наука зробила велетенський стрибок вперед: від прилавків і реклами у вигляді малюнків на вітринах виконаних вручну до маркетингу у всесвітній мережі, у якій будь-яка людина будь-коли і будь-де може побачити рекламну інформацію, яка необхідна комерційним структурам. Багатьом людям вже відомо що у наші часи збір інформації прийняв такі масштаби, що важко уникнути збору тих даних, які б ви не хотіли давати нікому. «Інформаційний слід» від людей залишається всюди і потім використовується для просування товарів та послуг у напрямку схожої на вас аудиторії.

На сьогоднішній день компанії, які не використовують розумну контекстну рекламу та не проводять поглиблений аналіз своєї стратегії продаж, не є конкурентоспроможними. Тому виникла необхідність у використанні інструментів, які допомогають у конкурентній боротьбі.

Аналізуючи дані у відкритому доступі та у своєму розпорядженні, компанії намагаються дізнатися оптимальні налаштування своєї комерційної діяльності. Один з найважливіших факторів – це аналіз своїх цін та оптимізація маркетингу.

Даний дипломний проект створений з метою допомогти комерційним компаніям знайти «точку дотику» з попитом клієнтів, оптимізувати маржу та проаналізувати роботу платформ розміщення продуктів задля максимальної прибутковості.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ

1.1. Аналіз проблем

Проблемною областю, яка потребує аналізу і оптимізації є потреби користувачів. Зазвичай, коли компанії хочуть задати оптимальну ціну для своїх товарів, вони дивляться на товари своїх конкурентів або намагаються оцінити товар своїми методами. При цьому, вони далеко не завжди можуть визначити реальну ціну, яка буде співпадати з рівнем потреб користувачів та максимальною вигодою від продажу продуктів.

Інтернет-магазини використовують різних рекламодавців для реклами своїх товарів, але для конкретної компанії їх ефективність різна. Окрім того, якщо замовити послуги декількох компаній, можна залучити більшу кількість покупців, проте ця база покупців не сумується, і хоч оплата частіше за все проходить за кожен перехід на сторінку, доводиться використовувати дорожчі варіанти рекламних платформ для додаткових продажів. Реальна користь кожної такої платформи не є очевидною, оскільки вони по суті є конкурентами і активуються одночасно.

Дуже важливими подіями для компаній є нестандартні ситуації продажів: акції, знижки на свята та будь-які інші нестандартні маніпуляції з цінами. Коли будь-яка компанія чи інтернет-магазин хочуть влаштувати подібні заходи, їм необхідно розуміти, як це буде виглядати і які фінансові наслідки понесе фірма, але це дуже важко визначити через складність вилучення інформації з даних тривіальними методами, що призводить до фінансово неефективних кроків і нерозумної цінової політики у таких ситуаціях.

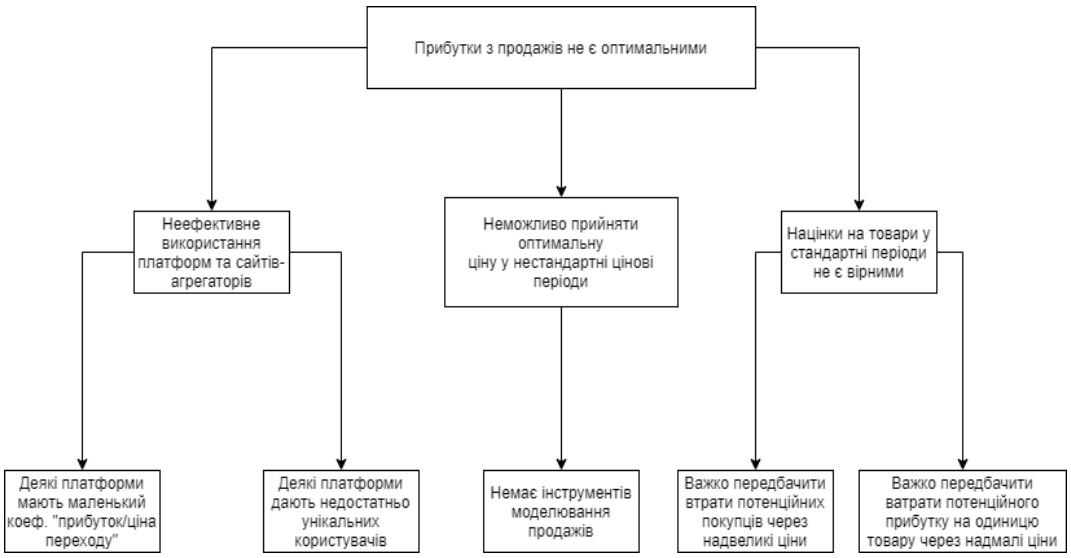


Рис. 1. Дерево проблем

1.2. Опис рішень

Під час розробки даного дипломного проекту були проаналізовані:

1) Алгоритм оцінки ефективності платформ для розміщення реклами товарів в інтернеті, від якого залежить аналіз відношення продажів до переглядів відносно кожної платформи;
2) Алгоритм оцінки впливу маржі на кількість унікальних переглядів сторінки;
3) Результати зміни параметрів торгівельного підприємства (фактичний вплив на продажі в залежності від зміни націнки та конфігурації системи)

Одне з найкращих рішень у цьому випадку – аналіз ринку з точки зору ефективності цінової політики за допомогою збережених даних про продажі.

Цей аналіз дозволяє визначити фактичний ціновий рівень і дає інформацію про компанію та її положення на ринку незалежно від інших комерційних суб’єктів, які є по суті конкурентами компанії серед користувачів даного продукту.

Важливим моментом є моделювання продажів компанії при певному рівні маржі або переглядів: це допомогає не тільки визначити найкращу маржу на продукти компанії, але й передбачає її прибутки при надзвичайному стані ринку або цінової політики (наприклад, розпродажі чорної п'ятниці або нового року).

Однією з вимог формування оптимальної ціни є аналіз ефективності рекламних платформ та прайс-агрегаторів. Платформи та прайс-агрегатори дуже схожі, але між ними є відмінності. Зазвичай стандартні рекламні платформи, такі як Google Ads використовують контекст користувачів – інформацію про їх інтереси, базовану на "цифровому сліді", пошуках та ін. Далі цей самий контекст визначає для кожного користувача рекламу, яка буде показана при завантаженні користувачем сторінок з контентом рекламодавців.

Прайс-агрегатори – це модифікований тип рекламних платформ. Вони аналізують ціни продуктів, які компанії-продавці вирішили виставити на продаж, та впорядковують їх, надаючи перевагу тим продуктам компаній, які мають найбільш вигідну для покупця ціну. У цьому випадку ситуація дуже схожа зі звичайними рекламними платформами, але маржа суттєво впливає на результуюче представлення продуктів потенційному покупцю.

Результативність реклами на певних платформах важлива для аналізу комерційної стратегії підприємства незалежно від того, є це класична рекламна платформа чи сайт-агрегатор цін. Якщо проаналізувати загальний шлях користувача до покупки товару, то можна побачити наступні кроки:

1) Перехід на сторінку партнера рекламодавця або landing page – користувач мережі інтернет переходить на будь-яку веб-сторінку, яка у своєму вмісті має банер чи інший рекламний контент представлений рекламною платформою, яка надає зацікавленій у продажі стороні замовлені в неї рекламні послуги. Користувач перейшовший на дану веб-сторінку бачить рекламний контент і приймає рішення про можливий перехід на сторінку рекламної платформи. На даному етапі

велику роль відіграють точність контекстної реклами та оформлення банеру (чи іншого представлення). Крім того, доволі часто рекламодавці прописують ціну у самому банері для приваблення користувача. Це дозволяє не переходити на сторінку платформи, щоб побачити реальну ціну. У цьому випадку ключову роль грає маржа, яка була виставлена компанією-продавцем. Можна зазначити, що крім традиційних банерів часто використовуються так звані *landing page* або *сторінки-вітрини*, які є повністю рекламними. Але це вимагає додаткової роботи над дизайном і часто відображають цифри, у нашому випадку-ціну.

2) Перегляд товару на сторінці рекламодавця - користувач переходить на сторінку рекламодавця (платформу чи прайс-агрегатор) і бачить опис товару та інформацію пов'язану з ним. Серед представленої інформації є ціна (у агрегаторах - при переході), власне опис, а також інші рекомендовані товари. На даному етапі оформлення сайту рекламодавцем сильно впливає на процес, але вплив маржі на рішення користувача є більшим. Якщо ціна надто велика, то користувач менш ймовірно придбає товар. На багатьох сайтах-агрегаторах цін у кожної сторінки товару є кнопка "порівняти", яка дозволяє користувачеві миттєво отримати список тих самих товарів від кожного інтернет-магазину, який виставив аналогічний товар. Порівнявши ціни, можливий покупець зазвичай обирає оптимальний варіант з фінансової точки зору, оскільки функціонал видає ідентичні товари. Більше того, якщо ціна на інші товари виставлена вірно, то, можливо, вони будуть висвітлені у зоні рекомендацій. Платформи, і тим більш прайс-агрегатори мають інструменти для розумного пошуку та порівняння цін, що допомагає користувачам визначитись з найвигіднішими цінами, тому розумна маржа допомагає потенційним покупцям знайти товари даного інтернет - магазину безпосередньо з сайту платформи.
3) Додавання товару до кошика – сайти-агрегатори не направляють користувача на сторінку продажу прямо. Кожна сторінка самого товару

вказує проміжок цін і при переході на сторінку порівняння переадресує користувача на сторінку товару на сайті інтернет-магазину чи виробника, але у цьому випадку користувач майже завжди додає товар до кошика вже на кінцевому сайті, оскільки майже всю інформацію було подано на агрегаторі. Коли користувач додав товар у кошик, роль ціни і рекламної платформи закінчена.

Виходячи з цього ланцюга переходів, можна впевнено сказати, що вибір рекламної платформи чи сайта-агрегатора має значення, тому необхідно враховувати їх як один з факторів, які впливають на продажі. Іншим фактором можна визначити ціни, від яких залежить рішення покупця не тільки про те, чи треба купувати товар, але й де його купувати.

Таким чином, збираючи дані з продажів та рекламних платформ, можна побудувати модель, яка допоможе передбачати продажі у стандартних та нестандартних ситуаціях, а також визначити ефективність використання платформ, на яких рекламуються товари.

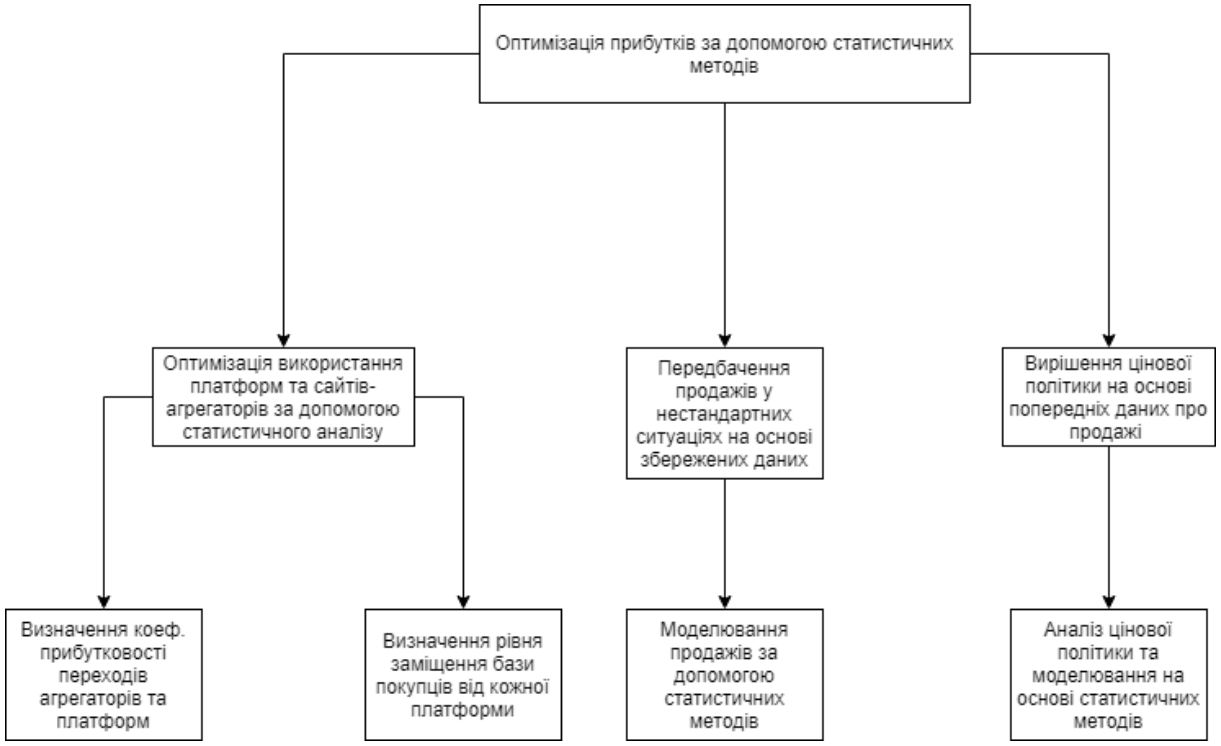


Рис. 2. Дерево рішень

1.3. Огляд існуючого програмного забезпечення

В дипломному проекті були досліджені та проаналізовані можливості найпоширеніших існуючих програмних засобів. Метою проведеного огляду є дослідження існуючих програмних продуктів, які призначені для автоматизації збору та аналізу продажів інтернет-магазинів та їх маржинальної політики.

Таблиця 1

Порівняння з аналогами

Система управління	Seller Republic	Blueboard .io	PriceLab	Minderest
Популярність за версією google.com	8	6	6	9
Аналіз прибутковості платформ	–	–	–	+
Моделювання продаж	–	–	–	–
ПЗ універсальне відносно рекламних платформ	–	+	+	+

Був проведений аналіз особливостей кожного з продуктів.

Seller Republic:

- Переваги:
  1) Пробний період – 14 днів.
  2) Додаток легко встановити, оскільки він по суті є прикладом платформо - орієнтованого ПЗ і швидко розгортається на Amazon Web Services.
  3) Додаток орієнтований на Amazon і тому повністю підтримує його API, таким чином він може автоматично змінювати ціни товарів за допомогою вбудованого функціоналу, не потребує додаткової роботи для автоматичного використання результатів.

4) Додаткове регулювання цінової політики на платформі, дає змогу налаштувати продажі під свій смак і нестандартні ринкові ситуації.
5) Вбудовані цінові політики та стратегії продажів дають можливість покращити показники на даній платформі.

- Недоліки:

1) Система орієнтована під певну платформу, а саме під Amazon, це взагалі позбавляє можливості використовувати рішення для інших платформ, а тим більше порівнювати з ними даний сайт для розміщення товарів. Крім того є багато регіонів без філіалів Amazon, що виключає можливість використовувати платформу, для виставлення своїх товарів у даних локаціях.
2) Система працює у реальному часі та не дає можливості проводити моделювання продажів, хоч і збирає дані про продажі та конкурентів.

Blueboard.io:

- Переваги:

1) Зручний функціонал по роботі з відгуками.
2) Розширена підтримка сповіщень на різні девайси, у цей функціонал входять сповіщення про відгуки, асортимент (відсутність товарів), тощо.
3) Будь-яка кількість користувачів одним аккаунтом

- Недоліки:

1) Відсутність можливості моделювання продажів даного інтернет-ретейлеру
2) Відсутність аналізу прибутковості платформ

PriceLab:

- Переваги:

1) Можливість виявити порушення MAP (найменшої пропонованої ціни), що може казати про підозрілі джерела товару або порушення домовленості про найменшу ціну. Система автоматично зберігає скріншоти та допомагає виявити підозрілі товари.
2) Порівняння цін з іншими продавцями на платформах та зручне їх відображення на інформаційній панелі .
3) Додаткові поради щодо виставлених цін або авторедагування цін на платформах самим ПЗ.

- Недоліки:
  1) Відсутність моделювання продажів.
  2) Відсутність аналізу прибутковості платформ.

Minderest:

- Переваги:
  1) Мобільний додаток.
  2) Керування товарами розташованими на декількох платформах, уникнення «цінових війн».
  3) Інтерфейс моніторингу цін на рекламних платформах чи сайтах-агрегаторах цін.
  4) Автоматичні скріншоти аномальних змін в цінах товарів.
  5) Зручний плагін для візуалізації даних про товари підчас перегляду платформ.
  6) Розширений функціонал рекомендацій популярних товарів для продажу.
- Недоліки:
  1) Моделювання продажів неможливе.
  2) Відсутній функціонал аналізу прибутковості рекламних платформ чи сайтів-агрегаторів цін, хоч і є можливість контролювати продукти на різних платформах.

3) Керування продажів на декількох платформах не є універсальним і виконується на платформах, які підтримує продукт.

1.4. Актуальність розробки

Були досліджені чотири різні системи управління рекламою, до яких є доступ в мережі Internet*:*

1) Seller Republic;
2) Blueboard.io;
3) PriceLab;
4) Minderest.

Як видно з порівняльного аналізу (табл. 1), найбільший спектр можливостей для аналізу ефективності цінової та маржинальної політики у «Minderest». Аналіз ресурсів інтернету, які спрямовані на порівняння програмних рішень для компаній показав, що Minderest має високий рейтниг, в нього багато сильних клієнтів, таких як L'Oreal та Sony. Самий функціонал цього ПЗ дозволяє проводити дуже впенені операції на декількох платформах та отримувати зворотній відгук, який включає в себе рекомендації по ціні, їх автоматичне регулювання. ПЗ навіть пропонує популярні продукти, які б було непогано виставляти на платформах. Однак Minderest проводить аналіз у реальному часу і не допомагає створити модель продажів з заданими даними, а можливість керування ціновою політикою та аналіз продажів на декількох платформах (зі списку платформ, які підтримує ПЗ) хоч і є ефективними інструментами, проте не дають змогу оцінити прибутковість платформ, тим більше в умовах малого бізнесу і в різних регіонах.

В результаті проведеного порівняння функціоналу основних програмних продуктів за обраними параметрами можна дійти висновку, що жоден з існуючих програмних продуктів не задовольняє вимоги продавця в

повній мірі. Хоч реалізації і є дуже багатофункціональними, але підхід зазвичай залежить від цін інших компаній і не дає змогу передбачити поведінку продажів у нестандартній ситуації. Немає функціоналу з оцінки прибутковості платформ та заміщення бази покупців, рішення направлені на великий бізнес, який буде розташовувати продукти на багатьох платформах і може пожертвувати грошима на рекламу.

Таким чином проблема є актуальною і рішення має можливість зайняти свою позицію на ринку.

2. ПРОВЕДЕННЯ ДОСЛІДЖЕННЯ МОЖЛИВОСТІ ЗАСТОСУВАНЯ СТАТИСТИЧНИХ МЕТОДІВ ДЛЯ ВИРІШЕННЯ ЗАДАЧІ

2.1. Існуючі методи аналізу економіки інтернет-підприємств

Існують різні методи оцінки роздрібної чи оптової ціни товарів комерційної структури, які йдуть на продаж. Проте важко сказати який з методів є оптимальним оскільки відрізняються умови у яких перебуває дана економічна система. Це стосується і даних доступних для аналізу статистичними методами.

На практиці велику кількість методів обробки і аналізу даних важко використати через малий розмір магазинів і обмеженість ресурсів. Це призводить до результатів, які не відповідають дійсності. Тому для вирішення задачі отримання аналізу даних на основі малої вибірки необхідно використовувати особливі методи.

Створення загальної картини про фінансову доцільність компанії як у плані націнки на товари, так і у виборі інших параметрів (конфігурація набору платформ для реклами товарів на продаж) є основною метою використання статистичних методів обробки інформації. Сайти-агрегатори цін дозволяють порівнювати пропозиції даної компанії з іншими виробниками чи продавцями. Це підвищує залежність остаточної кількості продаж від націнки на товари.

Оцінкою можливих шляхів формування оптимальної цінової та маркетингової стратегії займалися вчені Н. І. Верхоглядова [1], С. М. Дайновська [2], В. Л. Корінєв [3], В. С. Пінішко [4], Л. О. Шкварчук [5]. Вони розглядали питання ціноутворення класичних торгівельних підприємств без використання інформаційних технологій для отримання переваги на ринку.

Ю. В. Карпенко [6] дослідив шляхи потенціального покращення результатів торгівлі саме підприємств, які займаються комерцією за допомогою всесвітньої мережі Інтернет, відокремивши їх від класичних комерційних структур у плані підходу до становлення цінової політики.

Зазначені методи економічного аналізу зводяться до академічного підходу з точки зору фінансово-маркетингового аналізу. В цій роботі не розглядається можливість покращення показників роботи компанії шляхом обробки і аналізу даних, які зберігаються в їх базах даних. Звіти комерційної діяльності за квартал не дозволяють розрахувати точний алгоритм діяльності системи при різних налаштуваннях параметрів керування комерційною структурою, особливо при коротких термінах та мінімальних об’ємах даних. Таким чином, питання можливих алгоритмів для дослідження і оптимізації інтернет-торгівлі за допомогою програмного забезпечення залишається відкритим.

2.2. Формування мети дослідження

Метою даної наукової роботи є дослідження можливості створення аналітичного програмного забезпечення на основі маржинальних флуктуацій – невеликих випадкових змін в ціні. Це може дозволити оцінити коректність маржинальної політики та створити відповідну математичну модель використавши найменший об’єм даних та мінімум часу. Використання такого ПЗ дозволить спрогнозувати економічну поведінку торгівельного підприємства, яке використовує мережу Інтернет як головне джерело клієнтської бази та механізм продажу своїх товарів.

2.3. Логіка вибору шляхів рішення задачі

Першим кроком для дослідження даного питання є аналіз типового набору даних, який реєструється у будь-якій комерційній структурі, яка пов’язана зі всесвітньою мережею, оскільки архітектури кожної електронної системи, яка підтримує роботу підприємства відрізняються: у багатьох випадках типи баз даних, схеми та налаштуваннях будуть не схожі між собою. Це дуже важливо через те, що використовуючи спеціалізовані дані, які зберігаються у дуже малої кількості підприємств, програмне забезпечення не буде показувати результати, які стосуються малої множини подібних економічних структур, або взагалі не можуть бути відтворені. Прикладом

таких даних є «стан товару» інтернет-магазину Ebay, він може приймати значення: «новий», «новий інший (див. деталі)», «відновлений на заводі», «відновлений продавцем», «бувший у вжитку», «на запчастини чи неробочий». Облік цієї властивості ведуть доволі мало інтернет-магазинів, оскільки можливі великі похибки і практика недоцільна, якщо на ресурсі товар продається не користувачем-продавцем. Але цей параметр у поєднанні з типом товару може мати потенціальну кореляцію з ціною за якою користувачі ресурсу згодні купити виставлений виріб.

Таким чином необхідно обрати дані, які зберігаються у абсолютній більшості торгових інтернет-ресурсів, серед таких даних можна виділити звіти продажів, оскільки у підприємств, які займаються збутом товарів, повинна зберігатися інформація про код товару, його ціна в момент продажу, та час самої покупки. Для адекватного комерційного процесу зберігається також націнка на товар, сам цей параметр дозволяє керувати остаточним рівнем продажів та прибутку з одиниці товару. Проте, структура контролю маржі у різних фірмах зазвичай різна і вона частіше за все залежить від виробів, які йдуть на продаж. Наприклад, у магазинах загального призначення у кожного товару може бути різна націнка, проте якщо брати комерційні структури, які реалізують продажі електронної продукції в Інтернет, то доволі часто маржа співпадає на усіх виробах. Це є найпростішим випадком маржинальної стратегії підприємства, що може бути гарним прикладом для аналізу. У даному дослідженні проаналізована саме ця стратегія формування націнки товарів, яка стосується підприємств, які реалізують електроніку у всесвітній мережі Інтернет.

Наступним кроком для проведення дослідження комерційної діяльності є аналіз властивостей економічної системи, на які впливає націнка та висунення гіпотез про результуючу поведінку. У випадку торгівельних підприємств, які займаються комерційною діяльністю в Інтернет, покупці контактують з одиницями товару через контекстну рекламу чи шукаючи необхідні вироби безпосередньо на рекламних ресурсах – рекламних

платформах чи сайтах-агрегаторах цін. У другому випадку (прайс-агрегаторах) ціна впливає більше, так як перехід до реального лоту відбувається через список інших ресурсів, на яких виставлений цей самий товар, тому покупець може бачити ціни та порівняти усі платформи для продажу (і можливо обере товар виходячи з надійності продавця). Умисною буде гіпотеза, що на сайтах-агрегаторах кореляція цін з остаточним вибором буде сильнішою через очевидну для користувача конкуренцію в плані ціни і бренда продавця, тому оберемо цей тип рекламного ресурсу для статистичного аналізу.

2.4. Формування моделі

Остаточна модель повинна мати у собі глобальну націнку на товари підприємства (у даному випадку воно реалізує продажі електронних виробів), кількість переглядів кожної рекламної платформи (у цьому прикладі, це витрати на рекламу, що еквівалентно переглядам, так як з підприємств сайти-агрегатори цін отримують плату за кожен перегляд користувачем) і результуючу кількість продажів за день. Бажано зберігати поточний стан кожного постачальника реклами, проте якщо зазвичай у торгівельній компанії достатньо багато покупців, можна позначати вимкнений стан як 0.

Отже, можемо зобразити кожне торгівельне підприємство на вертикальній осі, а рекламні платформи на горизонтальній. Узявши реальні дані по 3 магазинам, які використовують 4 прайс-агрегатори кожний, маємо наступний результат:

Таблиця 2

Параметри математичної моделі

	Price Ua	Nadavi	Hotline	Google Buy	Маржа (%)	Продажі
Mag$_1$	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}	Y_1
Mag$_2$	X_{21}	X_{22}	X_{23}	X_{24}	X_{25}	Y_2
Mag$_3$	X_{31}	X_{32}	X_{33}	X_{34}	X_{35}	Y_3

Кожна незалежна змінна X_{ij} має , де і – номер підприємства, а «j» - номер прайс-агрегатора. Y_i – залежне значення, це кількість продажів торгівельного підприємства за період кожного звіту.

Важко оцінити прибутковість маржинальної політики при сталій маржі на товари – результат її зміни не може бути відомим, поки фактична ціна не набуде іншого значення. Тому статистичний аналіз маржинальної політики реальної економічної системи зі сталою ціною потребує дуже великої кількості даних. Гіпотетично потрібним математичним прийомом у даному випадку можуть слугувати цінові флуктуації – зміни цін при кожному звіті. Таким чином залежність продажів від націнки буде оцінити легше при меншій кількості даних.

Крім того важливо оцінювати кількість витрат на рекламу, оскільки вони пропорціональні фактичній кількості переглядів, які корелюють із кількістю продажів.

2.5. Опис використання моделі та статистичних методів

Тепер, коли формування моделі виконано, необхідно обрати мову програмування та необхідні програмні засоби для реалізації аналізу

маржинальної стратегії – гарним інструментом для виконання поставленої задачі є мова програмування Python. Вона достатньо розповсюджена і часто використовується для виконання задач пов’язаних з математикою, статистикою та машинним навчанням. На її основі розроблені та протестовані усі необхідні програмні засоби для реалізації задач статистичного аналізу даних. Математичний функціонал був обраний з програмного модуля sklearn, програмний фунціонал для роботи з даними з pandas, а модуль для візуалізації даних – matplotlib.

Разом з цим, обрано коректну методику статистичного аналізу математичної моделі даного торгівельного підприємства. Оскільки однією з вимог до процесу дослідження є максимальний результат при малій кількості даних, то найкращим кандидатом є лінійна регресія через відсутність потреби у великій кількості даних. Проте, якщо зв’язок результуючої змінної з вхідними даними не буде функцією наближеної до деякої пропорції, цей алгоритм не є доцільним і необхідно використовувати методи для відтворення нелінійних залежностей, наприклад, поліноміальну регресію. У статистично-математичному програмному пакеті sklearn лінійна регресія реалізована класом sklearn.linear_model.LinearRegression у реалізації звичайного методу найменших квадратів.

Розмірність трьох масивів даних після приведення до єдиного формату: 138x6, де індекси записів співпадають з днями звітів у послідовності, таким чином дані упорядковані за зростанням у часі.

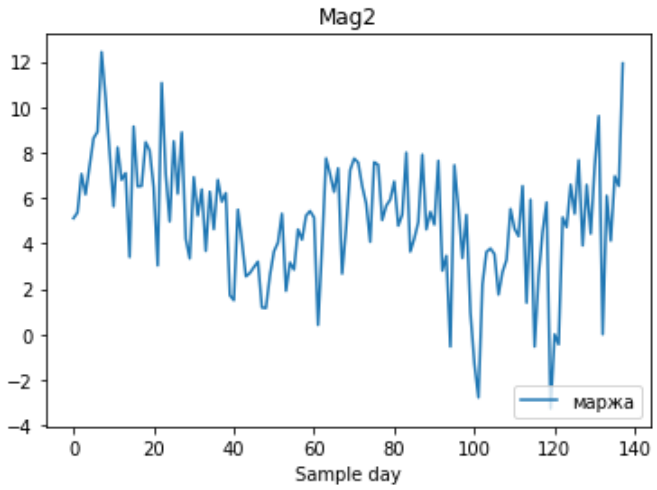


Рис 3. динаміка маржі у відсотках Mag2

Для уточнення моделі необхідно додати “затримку та різницю” часового ряду – параметр, який включає в себе значення попереднього дня та різницю між значенням його і поточного днів, таким чином будуть враховані дані за попередній день, таким чином маємо дані у форматі 138x18. Далі поділимо дані на частину для тренування і частину для перевірки результатів обравши 50 останніх днів для перевірки, таким чином маємо масиви 88x18 та 50x18.

Використаємо метод LinearRegression для пошуку лінійних залежностей моделі. Спочатку спробуємо провести лінійну регресію використовуючи тільки націнку на товари:

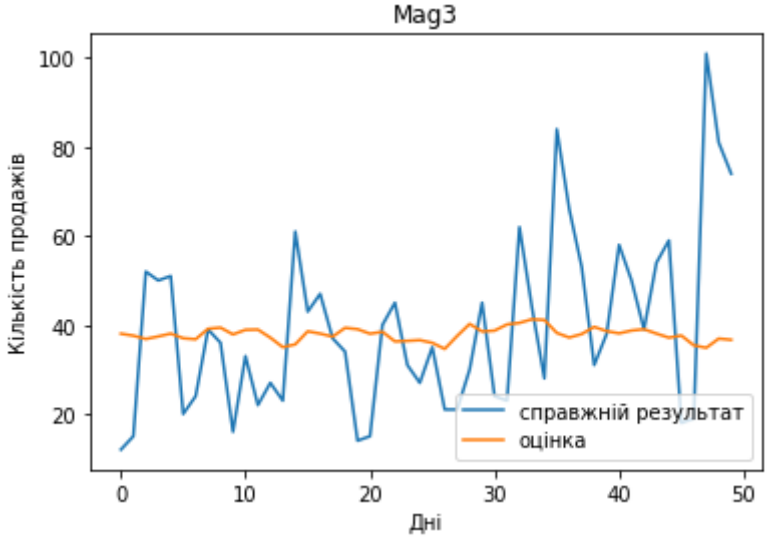


Рис 4. оцінка продажів за допомогою лінійної регресії на основі маржі

Числа над графіком – оцінка R^2 – це так званий коефіцієнт детермінації, який при максимальній кореляції залежної змінної від незалежної набуває одиниці, якщо результат регресії більш віддалений від фактичного значення ніж середнє арифметичне, то оцінка набуває негативного знаку.

Таким чином, результат регресії не корелює з реальними даними. В цьому випадку необхідно використати інші параметри чи методи. Спробуємо використати клас sklearn.preprocessing.PolynomialFeatures. Він приводить дані у вигляд, необхідний для реалізації аналізу нелінійних залежностей за допомогою sklearn.linear_model.LinearRegression.

Маємо результат при 2 степені полінома регресії:

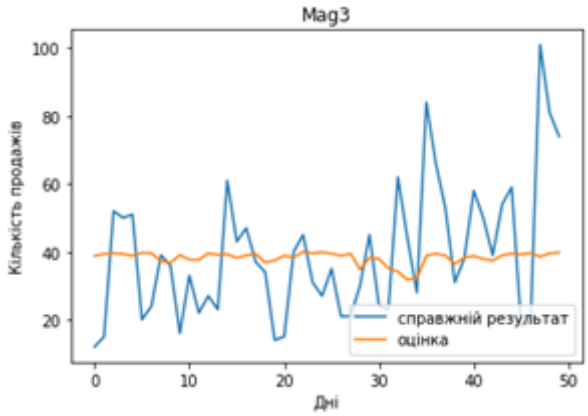


Рис 5. оцінка продажів за допомогою поліноміальної регресії степені 2 на основі маржі

Оцінка R^2 не набуває значення більше нуля, тому не має абсолютно ніякої передбачальної сили.

Спробуємо отримати аналогічні результати використовуючи маржу та витрати магазинів (які важко оцінити наперед) як незалежні змінні:

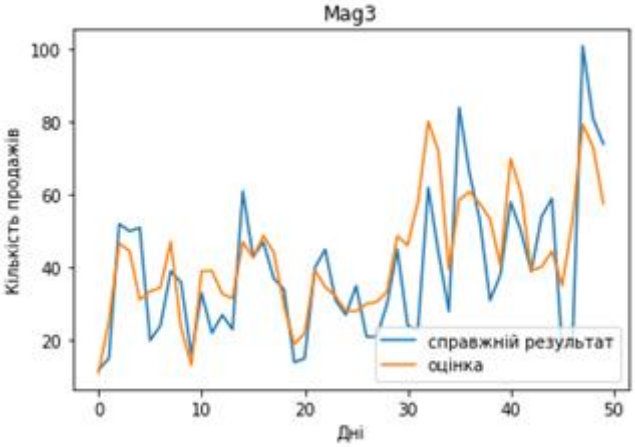


Рис 6. оцінка продажів за допомогою лінійної регресії на основі маржі та витрат на реламні платформи

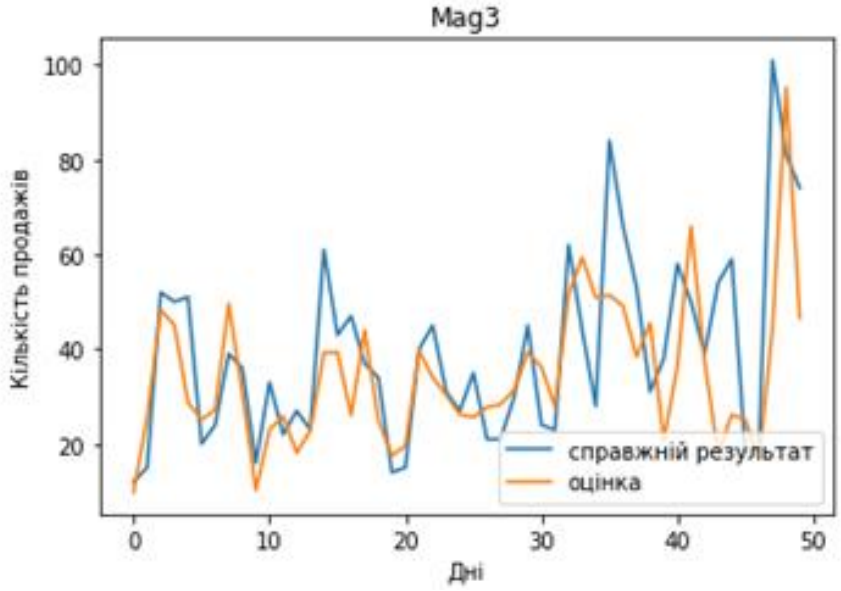


Рис 7. оцінка продажів за допомогою поліноміальної регресії степені 2 на основі маржі та витрат на рекламні платформи

Виходячи з отриманих результатів очевидно, що модель у даному вигляді неможливо використовувати для передбачення наслідків зміни маржі та для її оптимізації. Серед причин які призводять до такого: невідомі фактори ринку, малий об’єм даних, слабка залежність кількості продаж від малих флуктуацій ціни. Таким чином все ж існує ймовірність, що флуктуації ціни в межах 1-10% можуть давати уточнення оцінки залежності, але якщо вони не застосовуються до товарів декілька років, можливий періодичний вплив на продажі в залежності від дати чи сезону, цей вплив неможливо дослідити при малих термінах, тому, зважаючи на можливі втрати через випадкове регулювання націнки, вплив даної методики на підприємство негативний. Можливо, необхідно вдосконалити метод флуктуацій, адже не факт, що абсолютно випадкові зміни – найкращий шлях дослідження впливу маржі.

Отже, програмному забезпеченню, яке оперує статистичними методами аналізу, необхідні якісно інші дані та за більший проміжок часу, це ставить під сумнів ефективність маржинальних флуктуацій як показника впливу маржі на продажі через непередбачені фінансові наслідки. Є ймовірність, що метод

флуктуацій можливо вдосконалити для покращення вхідних даних, проте на даний момент ПЗ для статистичного аналізу маржинальної політики підприємства повинне опиратися на більш класичні методи.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ СТАТИСТИЧНИХ МЕТОДІВ ТА МАТЕМАТИЧНИХ МОДЕЛЕЙ

3.1. Програмний інструментарій для дослідження і реалізації

Для статистичного дослідження та аналізу властивостей економічної моделі був використаний веб-додаток Jupyter Notebook, програмним ядром якого є IPython – інтерактивна оболонка для мови програмування Python, яка вводить в користування так звані «клітини» (з англ. «cell») у кожній з яких можливо проводити виконання коду на мові Python. Обгортка надає можливість у будь-який час виконувати код у вказаній клітинці і таким чином можливо запускати частини програми у довільній послідовності, що сильно пришвидшує дослідницьку діяльність, адже необхідно оновлювати та запускати заново лише певну частину коду, усі змінні залишаються у пам’яті постійно. Крім того цей програмний продукт надає можливість використовувати так звану «клітинну магію» - унікальний функціонал обгортки, який доповнює властивості самої мови програмування шляхом введення додаткового символьного синтаксису і обробки контексту. Самий Jupyter Notebook надає можливість візуалізувати вивід мови Python знизу від клітини.

Для створення економічної моделі підприємства необхідне створення спеціалізованого програмного забезпечення, орієнтованого на аналіз і прогнозування результатів зміни таких параметрів як націнка і кількість витрат на рекламу. Реалізувати засоби моделювання можливо за допомогою засобів вище, які були використані для дослідження можливості статистичного аналізу маржинальної стратегії і ефективності використання рекламних платформ.

При створенні якісного програмного забезпечення необхідне ретельне планування архітектури. Не менш важливими факторам є й коректний вибір джерела та методів обробки даних.

Джерелом даних для даного програмного забезпечення можуть слугувати будь-які файли, з якими може працювати pandas. Модуль pandas реалізує представлення даних у вигляді так званих датафреймів (англ. dataframe), які мають вигляд таблиці. Самий модуль реалізує операції з таблицями, традиційні для реляційних баз даних, таким чином це є по суті програмне забезпечення, оперуючи схоже на звичайні СУБД. Таким чином серед файлів які бібліотека підтримує напряму, є TSV та CSV. Ці два формати є одними з найпопулярніших, проте частіше зустрічається саме CSV як формат експорту даних з фінансового або аналітичного програмного забезпечення. Таким чином, найкраще імпортувати дані для дослідження саме з файлів формату “.csv”.

Лістинг 1. Зчитування даних з файлу формату CSV за допомогою бібліотеки pandas

```
df_lexore = pd.read_csv("mag1.csv", sep=';')
df_mobitek = pd.read_csv("mag2.csv", sep=';')
df_emagaz = pd.read_csv("mag3.csv", sep=';')
cols = ["Order", "PriceUa", "Nadavi", "XT", "Google buy", "%"]
df_lexore = df_lexore.fillna(0)[cols]
df_mobitek = df_mobitek.fillna(0)[cols]
df_emagaz = df_emagaz.fillna(0)[cols]
```

Слід зазначити, що кожен магазин у такому випадку буде мати вигляд списку Python, де кожен елемент – список зі значеннями рядку pandas. Значення відповідають Табл. 1. Цей тип структур буде використовуватися далі.

3.2. Підготовка даних

Після імпорту даних у програму, необхідно впевнитися у коректності формату та відсутності некоректних записів:

Лістинг 2. Очистка даних усіх магазинів від некоректних записів зі збереженням синхронності

```
def sync_fix(vals1, vals2, vals3, positive):
    total_len = min(len(vals1), len(vals2), len(vals3))
    vals1 = [[i] + vals1[i] for i in range(0, total_len)]
    vals2 = [[i] + vals2[i] for i in range(0, total_len)]
    vals3 = [[i] + vals3[i] for i in range(0, total_len)]

    v1 = []
    v2 = []
    v3 = []

    curr_v1 = 0
    curr_v2 = 0
    curr_v3 = 0

    cv1 = []
    cv2 = []
    cv3 = []

    for val1 in vals1:
        cv1 = val1
        curr_v1 = int(val1[0])
        found = True
        for val2 in vals2:
            cv2 = val2
            curr_v2 = int(val2[0])
            if curr_v2 < curr_v1:
                continue
            if curr_v2 == curr_v1:
                break
            if curr_v2 > curr_v1:
                found = False
                break
        if found:
            for val3 in vals3:
                cv3 = val3
                curr_v3 = int(val3[0])
                if curr_v3 < curr_v1:
                    continue
                if curr_v3 == curr_v1:
                    break
```

```
                if curr_v3 > curr_v1:
                    found = False
                    break
            if found:
                v1.append(cv1[1:])
                v2.append(cv2[1:])
                v3.append(cv3[1:])
        return v1, v2, v3
```

Тут vals1, vals2, vals3 – списки зі значеннями про які було сказано вище, positive – очищення рядків, які містять значення з негативним знаком.

У випадку, якщо синхронізація непотрібна, можливо використати очищення магазинів у несинхронному порядку:

Лістинг 3. Функція очищення даних одного магазину без збереження синхронності записів (datafix)

```
def datafix(preX, positive):
    X = []
    k = (1 - int(positive)) * (-10000)
    for x in preX:
        frame = []
        valid = True
        for val in x:
            if val is float:
                if val < k:
                    valid = False
                    break
                else:
                    frame.append(val)
            else:
                try:
                    float(val)
                except:
                    try:
                        float(val.replace(',', '.'))
                    except:
                        valid = False
                        break
                    else:
                        v = float(val.replace(',', '.'))
```

```
                            if v < k:
                                valid = False
                                break
                            else:
                                frame.append(float(val.replace(',',
'.')))
                    else:
                        if (float(val) < k):
                            valid = False
                            break
                        frame.append(float(val))
            if valid:
                X.append(frame)
        return X
```

Через те що доступні дані трьох магазинів зі значеннями синхронізованими по дням, і так як ці дані включають в себе стан користування рекламними платформами, за допомогою статистичних методів можливо «зав'язати» ці три магазина і змоделювати вплив одного на інший – так як вони мають той самий тип продукції – електроніку, тому корегування даних вище враховує цей факт і не допускає втрату синхронізації по дням. Проте таким чином відбувається додаткова втрати інформації і при виконанні оцінки впливу магазинів, зневажуючи конкуренцією між ними, можливо провести очищення з меншою кількістю видалених даних:

Лістинг 4. Процес використання методів очистки даних

```
    if sync:

                pre_mag1 = df_mag1.values.tolist()
                pre_mag2 = df_mag2.values.tolist()
                pre_mag3 = df_mag3.values.tolist()

                vals_mag1,  vals_mag2,  vals_mag3  =  sync_fix(pre_mag1,
pre_mag2, pre_mag3, POSITIVE)
            else:

                vals_mag1 = datafix(df_mag1.values.tolist(), POSITIVE)
                vals_mag2 = datafix(df_mag2.values.tolist(), POSITIVE)
```

```
vals_mag3 = datafix(df_mag3.values.tolist(), POSITIVE)
```

Тут sync – параметр, який має позитивне значення у випадку якщо дані необхідно тримати синхронізованими по дням.

3.3. Реалізація математичних моделей

Для створення правильної математичної моделі доволі важливе значення мають дані, які поступають на обробку статистичними методами. Функція наведена нижче виконує обробку масиву даних (1) і таким чином редагує набір параметрів. Функція, яка реалізує дану операцію з вхідними у модель даними:

Лістинг 5. Метод трансформації даних для утворення математичної моделі

```
def method_transform(method, X1, X2, X3):
    if method == 1: #1 ads_sum
        R1 = [[sum(x[:-1])] for x in X1]
        R2 = [[sum(x[:-1])] for x in X2]
        R3 = [[sum(x[:-1])] for x in X3]
        return R1, R2, R3
    if method == 2: #2 ads_array
        R1 = [x[:2] + [x[3]] for x in X1]
        R2 = [x[:3] for x in X2]
        R3 = [x[:3] for x in X3]
        return R1, R2, R3
    if method == 3: #3 [ads_sum, %]
        R1 = [[sum(x[:-1])] + [x[-1]] for x in X1]
        R2 = [[sum(x[:-1])] + [x[-1]] for x in X2]
        R3 = [[sum(x[:-1])] + [x[-1]] for x in X3]
        return R1, R2, R3
    if method == 4: #4 [ads_array] + [%]
        R1 = [x[:1] + [x[3]] + [x[-1]] for x in X1]
        R2 = [x[:3] + [x[-1]] for x in X2]
        R3 = [[x[0]] + [x[2]] + [x[-1]] for x in X3]
        return R1, R2, R3
    if method == 5: #5 %
        R1 = [[x[-1]] for x in X1]
```

```
        R2 = [[x[-1]] for x in X2]
        R3 = [[x[-1]] for x in X3]

        return R1, R2, R3

    if method == 6: #0 array

        return X1, X2, X3

    if method == 7:

        R1 = [x[:-1] for x in X1]

        R2 = [x[:-1] for x in X2]

        R3 = [x[:-1] for x in X3]

        return R1, R2, R3
```

Тут X1, X2, X3 – дані структури типу (1), method – метод обробки масивів даних. Зміст методів наступний:

1) Сума загальних витрат на рекламу усіх рекламних платформ.
2) Набір витрат на рекламні платформи, які були часто у використанні. Для X1 – “PriceUa” , “Nadavi”, “Google Buy”
   Для X2 – “Order”, “PriceUa”, “Nadavi”
   Для X3 – “Order”, “PriceUa”, “Nadavi”
3) Сума загальних витрат на рекламу усіх рекламних платформ та націнка на товари.
4) Набір усіх параметрів пункту 2, а також рівень маржі на товари кожного торгівельного підприємства відповідно.
5) Кожен з параметрів містить тільки рівень націнки на товари відповідного інтернет-магазину
6) Вхідні параметри містять набір значень витрати на рекламні платформи, а також націнку на товари кожного відповідного інтернет-магазину
7) Даний набір параметрів містить усі рекламні платформи, майже не наявні або які взагалі були виключеними під час усього періоду збору даних.

Серед методів обробки вхідних параметрів також можна зазначити метод затримки та різниці, так званий lag-diff. Його суть полягає у тому, що при обробці часових рядів, до усіх значень кожного запису додаються два додаткових параметри, які відповідають значенню попереднього дня та різниці між ним там наступним. Таким чином, кожен запис отримує «пам’ять» про попередній і таким чином має змогу отримувати додаткову аналітичну здібність, не тільки від налаштувань у даний час, але й від динаміки стану магазину.

Обробка даних зазначена вище може бути здійснена наступним кодом:

Лістниг 6. використання затримки та різниці для додання «пам’яті» часовому ряду

```
def lag_diff_features(X):
    Y = X
    feats = []
    if type(X[0]) is list:
        for i in range(0, len(X[0])):
            feats.append([x[i] for x in X])
    else:
        feats.append(X)
    lds = []
    for arr in feats:
        lds.extend(lag_diff(arr))
    for ld in lds:
        Y = combine_feats(Y, ld)
    return Y
```

Де lag_diff(X) має наступний зміст:

```
def lag_diff(X):
    res = []
    arr = [clear_len_1(x) for x in X]
#     print(arr)
    if arr[0] is not list:
        lag = [0]
        diff = [0]
        for i in range(0, len(arr)-1):
            lag.append(arr[i])
            diff.append(clear_len_1(arr[i+1])-clear_len_1(arr[i]))
        res.append(lag)
        res.append(diff)
    else:
            for j in range(0, len(arr[0])):
                lag = [0]
                diff = [0]
                for i in range(0, len(arr)-1):
                    lag.append(arr[i][j])
                                diff.append(clear_len_1(arr[i+1][j])-
                                            clear_len_1(arr[i][j]))
            res.append(lag)
            res.append(diff)
#     print(res)
    return res
```

3.4. Використання статистичних методів

Таким чином, після обрання одного з наявних методів перетворення масиву вхідних даних, необхідно скористатися статистичними методами для оцінки залежності результату – продажів інтернет-магазину від параметрів – витрат на рекламу чи націнки на товари (в залежності від обраного формату параметрів масиву вхідних даних). Код функції, яка реалізує цю операцію:

Лістинг 7. «Тренування» математичної моделі

```
def type_train(t_type, X1, Y1, X2, Y2, X3, Y3, degree=2):
    lg0 = linear_model.LinearRegression()
    lg1 = linear_model.LinearRegression()
    lg2 = linear_model.LinearRegression()

    if t_type == 1:

        lg0.fit(X1, Y1)
        lg1.fit(X2, Y2)
        lg2.fit(X3, Y3)

        return lg0, lg1, lg2

    if t_type == 2:
        X = [x1 + x2 + x3 for x1, x2, x3, in zip(X1, X2, X3)]

        lg0.fit(X, Y1)
        lg1.fit(X, Y2)
        lg2.fit(X, Y3)

        return lg0, lg1, lg2

p0 = make_pipeline(PolynomialFeatures(degree),
linear_model.LinearRegression())
p1 = make_pipeline(PolynomialFeatures(degree),
linear_model.LinearRegression())
p2 = make_pipeline(PolynomialFeatures(degree),
linear_model.LinearRegression())

    if t_type == 3:

        p0.fit(X1, Y1)
```

```
            p1.fit(X2, Y2)
            p2.fit(X3, Y3)

            return p0, p1, p2

        if t_type == 4:
            X = [[x1[0], x2[0], x3[0]] for x1, x2, x3, in zip(X1, X2, X3)]
            p0.fit(X, Y1)
            p1.fit(X, Y2)
            p2.fit(X, Y3)


            return p0, p1, p2
```

Параметри мають наступне значення:

1) t_type – тип методу способу використання статичтичних методів, від цього значення залежить факт залежності чи незалежності продажів конкретного інтернет-магазину від інших підприємств. Як було зазначено вище, дані у цьому дослідженні надані магазинами з однаковими типами товару – електронікою.
2) X1, X2, X3 – масиви з вхідними незалежними параметрами – витрати та маржа
3) Y1, Y2, Y3 – масиви з залежними параметрами – кількістю продажів інтернет-магазинів
4) degree – степінь полінома, який відображає залежність продажів від незалежних параметрів.

У даній функції першим чином аргументам lg0, lg1, lg2 присвоюються екземпляри класу “sklearn.linear_model.LinearRegression” (лінійна модель), він необхідний як для лінійної, так і для поліноміальної регресії. Далі, в залежності від обраного типу який задається вхідним параметром t_type, виконуються наступні дії:

1) Витрати на рекламні платформи та значення націнки на товари передаються методу lg_i.fit(), і таким чином наповнюють лінійну модель відповідними масивами незалежних та залежних від них даних.
2) Створюється один спільний масив даних X, який включає в себе усі незалежні дані інтернет-магазинів у змінних X1, X2, X3. Зберігаються дані по дням паралельно, тому кожен запис загального масиву даних (де кожен запис відповідає дню звіту) буде мати у собі значення усіх параметрів усіх магазинів в даний конкретний день. Даний масив буде використаний для лінійних моделей усіх інтернет-магазинів, проте у кожної з них будуть вдрізнятися значення Y_i. Таким чином кожна з моделей матиме різний результат при зазначених параметрах.
3) Перед цим та наступним параметром проводиться створення так званого пайплайну sklearn, його у даному модулі реалізує клас sklearn.pipeline, від автоматизує послідновне використання методів даної бібліотеки, дозволяючи реалізувати дану операцію простою послідовною передачею у make_pipeline() дій у вигляді вхідних аргументів. У даному випадку для кожного інтернет-магазину це здійснено наступним кодом:

Лістинг 8. Створення «пайплайну» статистичних методів

```
   p0            =            make_pipeline(PolynomialFeatures(degree),
linear_model.LinearRegression())
   p1            =            make_pipeline(PolynomialFeatures(degree),
linear_model.LinearRegression())
   p2            =            make_pipeline(PolynomialFeatures(degree),
linear_model.LinearRegression())
```

Виконання ціє функції відповідає наступному коду:

Лістинг 9. Реалізація роботи пайплайну

```
foo = PolynomialFeatures(degree)
X_ = foo.fit(X)
P = linear_model.LinearRegression(X_)
```

Таким чином кожній з зазначених змінних присвоюється об’єкт, у якого можливо викликати метод fit(X) і таким чином запустити ланцюг операцій. Після його виконання, у підсумку отримуємо результат так званої поліноміальної регресії. Це відбувається через спеціальну трансформацію параметрів, її виконує об’єкт PolynomialFeatures, як зазначалося раніше, саме від змінної degree, яка передається в нього і залежить степінь цієї трансформації і у свою чергу вид полінома після проведення регресії. При значенні 3 параметру t_type, вхідні незалежні змінні не об’єднуються і тому кількість продажів кожного інтернет-магазину буде залежати тільки від його стану.

У четверному випадку, також реалізується поліноміальна регресія, проте вхідні параметри моделі об’єднуються, тому кількість продажів певного магазину буде залежати не тільки від нього, але й від параметрів його конкурентів.

Лістинг 10. Функція оцінки рівня продажів

```
def type_predict(t_type, X1, Y1, X2, Y2, X3, Y3, degree=2):
    lg0 = linear_model.LinearRegression()
    lg1 = linear_model.LinearRegression()
    lg2 = linear_model.LinearRegression()

    Y1 = [[y] for y in Y1]
    Y2 = [[y] for y in Y2]
    Y3 = [[y] for y in Y3]

    if t_type == 1:

        X1_test = X1[-SAMPLES_COUNT:]
        X2_test = X2[-SAMPLES_COUNT:]
        X3_test = X3[-SAMPLES_COUNT:]

        X1 = X1[:-SAMPLES_COUNT]
        X2 = X2[:-SAMPLES_COUNT]
        X3 = X3[:-SAMPLES_COUNT]

        lg0.fit(X1, Y1)
```

```
            lg1.fit(X2, Y2)
            lg2.fit(X3, Y3)

            print(f'Mag1: {lg0.score(X1_test, Y1[-SAMPLES_COUNT:])}')
            print(f'Mag2: {lg1.score(X2_test, Y2[-SAMPLES_COUNT:])}')
            print(f'Mag3: {lg2.score(X3_test, Y3[-SAMPLES_COUNT:])}')


            R1 = lg0.predict(X1_test)
            R2 = lg1.predict(X2_test)
            R3 = lg2.predict(X3_test)

            return R1, R2, R3

        if t_type == 2:
            X = [x1 + x2 + x3 for x1, x2, x3, in zip(X1, X2, X3)]

            X_test = X[-SAMPLES_COUNT:]
            X = X[:-SAMPLES_COUNT]

            lg0.fit(X, Y1)
            lg1.fit(X, Y2)
            lg2.fit(X, Y3)

    print(f'Mag1: {lg0.score(X_test, Y1[-SAMPLES_COUNT:])}')
    print(f'Mag2: {lg1.score(X_test, Y2[-SAMPLES_COUNT:])}')
    print(f'Mag3: {lg2.score(X_test, Y3[-SAMPLES_COUNT:])}')

            R1 = lg0.predict(X_test)
            R2 = lg1.predict(X_test)
            R3 = lg2.predict(X_test)

            return R1, R2, R3

    p0          =          make_pipeline(PolynomialFeatures(degree),
linear_model.LinearRegression())
    p1          =          make_pipeline(PolynomialFeatures(degree),
linear_model.LinearRegression())
    p2          =          make_pipeline(PolynomialFeatures(degree),
linear_model.LinearRegression())

        if t_type == 3:
```

```
        X1_test = X1[-SAMPLES_COUNT:]
        X2_test = X2[-SAMPLES_COUNT:]
        X3_test = X3[-SAMPLES_COUNT:]

        X1 = X1[:-SAMPLES_COUNT]
        X2 = X2[:-SAMPLES_COUNT]
        X3 = X3[:-SAMPLES_COUNT]

        p0.fit(X1, Y1)
        p1.fit(X2, Y2)
        p2.fit(X3, Y3)

        print(f'Mag1: {p0.score(X1_test, Y1[-SAMPLES_COUNT:])}')
        print(f'Mag2: {p1.score(X2_test, Y2[-SAMPLES_COUNT:])}')
        print(f'Mag3: {p2.score(X3_test, Y3[-SAMPLES_COUNT:])}')

        R1 = p0.predict(X1_test)
        R2 = p1.predict(X2_test)
        R3 = p2.predict(X3_test)
        return R1, R2, R3

    if t_type == 4:
        X = [x1 + x2 + x3 for x1, x2, x3, in zip(X1, X2, X3)]

        X_test = X[-SAMPLES_COUNT:]
        X = X[:-SAMPLES_COUNT]

        p0.fit(X, Y1)
        p1.fit(X, Y2)
        p2.fit(X, Y3)

        print(f'Mag1: {p0.score(X_test, Y1[-SAMPLES_COUNT:])}')
        print(f'Mag2: {p1.score(X_test, Y2[-SAMPLES_COUNT:])}')
        print(f'Mag3: {p2.score(X_test, Y3[-SAMPLES_COUNT:])}')

        R1 = p0.predict(X_test)
        R2 = p1.predict(X_test)
        R3 = p2.predict(X_test)

        return R1, R2, R3
```

Дана функція має зміст схожий на type_train, проте по суті вона не повертає моделі відповідних інтернет-магазинів для подальшого використання, а проводить оцінку значень кількостей продажів підприємств усередині даного коду.

Ця частина програми використовується для разової оцінки певного масиву даних на відмінну від попереднього програмного компоненту, після виконання якого, отримані моделі можливо використовувати багаторазово, що можна буди побачити при виконанні дослідження питомої прибутковості витрат на рекламні платформи.

Перевагою даного використання математичних моделей, які реалізують статистичні методи, є автоматизація розбиття масиву значень відповідних параметрів на дві частини, де одна частина є тренувальною, а інша – тестовою. Тренувальна частина використовується для формування математичної моделі за допомогою регресії, тоді коли тестова використовується для оцінки точності роботи статистичних методів. Таким чином загальний масив розбивається, модель формується за допомогою даних у змінній X, які передаються у метод .fit(). Потім сформована модель проводить оцінку кількості продажів по дням, які не були використані для «тренування» математичної моделі і повертає результат по кожному інтернет магазину у вигляді масиву значень по кожному відповідному дню. Алгоритмічно в залежності від параметру t_type дії проводяться аналогічні до попередньої функції. Значення 1 – лінійна регресія та незалежні неконкурентні магазини, 2 – лінійна регресія, проте кількість продажів залежить від інших магазинів, 3 – поліноміальна регресія, продажі залежать тільки від параметрів відповідних магазинів, 4 – поліноміальна регресія, продажі залежать від параметрів усіх наявних магазинів.

Таким чином, маємо два різні програмні інструмента, один може бути використаний для оцінки часового ряду шляхом виділення тестової частини, інший можемо використовувати декілька разів та не обов’язково для оцінки тестового ряду.

Функцію «type_predict» можливо використати для задачі з оцінкою продажів за допомогою інформації про маржу, для цього необхідно реалізувати візуалізацію результатів та вхідних даних:

Лістинг 11. Функція зображення графіка

```
def draw_2_feats(feat1, feat2, f1_leg, f2_leg, title, y_label="Units"):
    line_chart1 = plt.plot(range(0,len(feat1)), feat1)
    line_chart2 = plt.plot(range(0,len(feat2)), feat2)
    plt.title(title)
    plt.xlabel('Дні')
    plt.ylabel(y_label)
    plt.legend([f1_leg, f2_leg], loc=4)
    plt.show()
```

Створивши обгортку даної функції можливо реалізувати конкретизацію використання draw_2_feats саме для результатів оцінки:

Лістинг 12. Зображення оцінки рівня продажів

```
def draw_sales(res, real, title):
    draw_2_feats(real, res, "справжній результат", "оцінка", title,
y_label="Кількість продажів")
```

Користуючись даним кодом, маємо:

Лістинг 13. створення математичної моделі та зображення результатів

```
deg=2
for t in range(1,4):
    for method in range(1,8):
        print(f"Method: {method}, type: {t}")
        M1, M2, M3 = method_transform(method, full_X1, full_X2, full_X3) #
Расходы для оценки продаж
               M1 = lag_diff_2(M1)
               M2 = lag_diff_2(M2)
               M3 = lag_diff_2(M3)
        res1, res2, res3 = type_predict(t, M1, Y1, M2, Y2, M3, Y3, deg)
               draw_sales(res3, Y3_test, 'Mag3')
```

Цей код виводить графік для кожного способу обробки масиву вхідних параметрів та для кожного статистичного методу. Приклад відображення такого графіка:

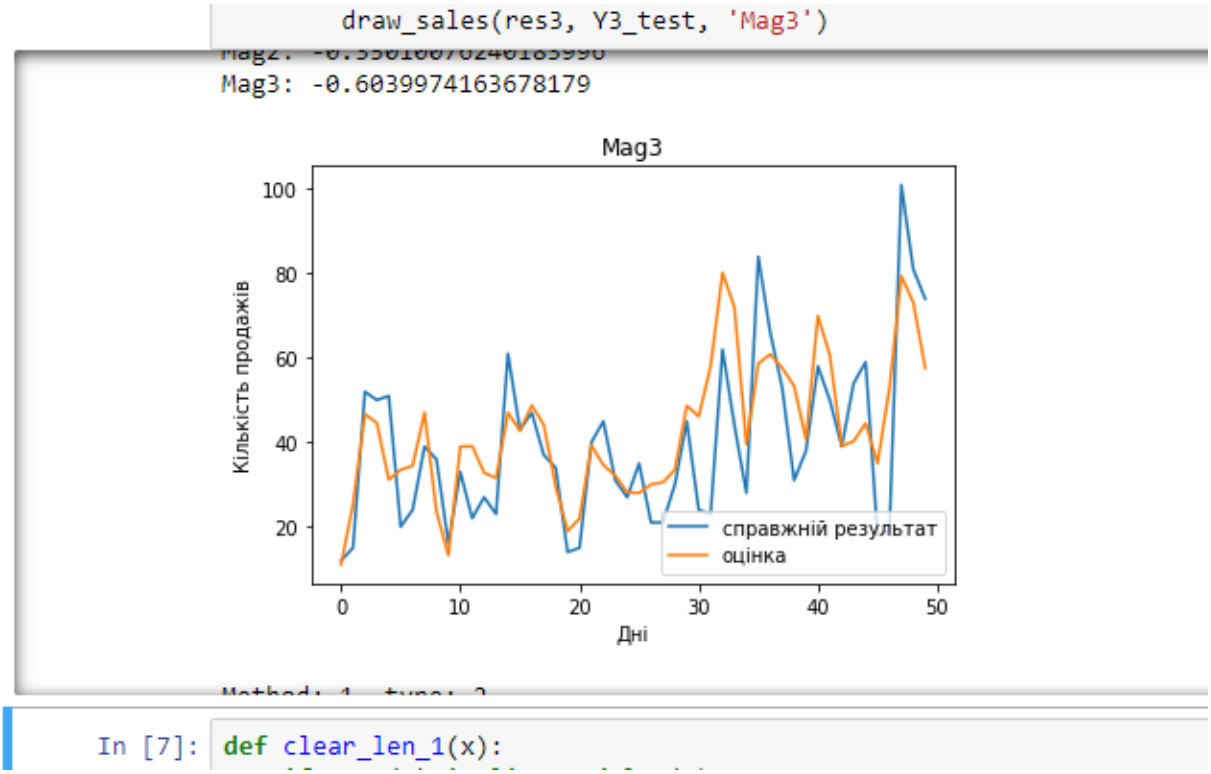


Рис 8. Відображення графіку залежності продажів від параметрів інтернет-магазину у Jupyter Notebook

Фунцію type_predict у даному випадку доцільніше використовувати для рішення інших задач. Однією з них є оцінка кільності продажів певного інтернет-магазину в залежності від витрат на платформи.

Цю оцінку можливо отримати, використавши type_train для побудови моделі на основі наявних даних. Ця модель буде мати поліном чи лінійний коефіцієнт в залежності від типу математичної моделі. Даний результат можна візуалізувати оцінивши деякий ряд від нуля до певного значення, при поліноміальній моделі, лінії оцінки будуть кривої форми, тоді как при лінійній – прямої.

Для виконання оцінки необхідно створити оцінку для кожної з платформ кожного магазину. Таким чином виникає деревовидна структура і для візуалізації результатів необхідна спеціальна функція.

Лістинг 14. Функція зображення словника результатів

```
def draw_dict(lines_dict, title, mar_res=False):
    plt.xlabel('Витрати на рекламу')
    plt.ylabel('Продажів')
    plt.title(title)
    legs = []
    if mar_res:
        legs = ["PriceUa", "Nadavi", "XT", "GoogleBuy"]
        for k, v in lines_dict.items():
            first = v[0]
            last = v[-1]
            plt.plot(range(0,len(v)), v)
        ks = [(v1-v0)/1200 for v0, v1 in zip(first, last)]
    else:
        for k, v in lines_dict.items():
            legs.append(k)
            plt.plot(range(0,len(v)), v)

    plt.legend(legs, loc=4)
    plt.show()
    if mar_res:
        k_max = max(ks)
        print("Начальні значення:\n")
        for leg, v0 in zip(legs, first):
            print("{:s}: {:.6f}".format(leg, v0))
        print("\nКоэф. эфективності:\n")
        for leg, k in zip(legs, ks):
            print("{:s}: {:.6f}".format(leg, k))
        print("\nВитрат на 1% маржи:\n")
        for leg, k in zip(legs, ks):
            print("{:s}: {:.6f}".format(leg, k*100))
        print("\nКоэф. Відносно найбільшого:\n")
        for leg, k in zip(legs, ks):
            print("{:s}: {:.6f}".format(leg, k/k_max))
```

Тут:

1) lines_dict – словник з масивами значень ліній, які необхідно візуалізувати
2) title – назва графіку
3) mar_res – булева змінна, яка визначає зміст lines_dict. Якщо вона має позитивне значення, обробка відбувається з особливостями рекламних платформ, інакше – як звичайного словника з лініями.

Будувати модель можна враховуючи залежність результату від інших магазинів та платформ, або незалежність від них. Для обох випадків необхідно створити два різних типи словників: незалежний і залежний.

Лістинг 15. Проведення аналізу рекламних платформ за допомогою незалежної моделі.

```
   lex_plot = {
       "PriceUA":  A1.predict(lag_diff_2([[x,  0,  0,  0]  for  x  in
list(range(0, 6000))])),
       "Nadavi":  A1.predict(lag_diff_2([[0,  x,  0,  0]  for  x  in
list(range(0, 6000))])),
       "XT": A1.predict(lag_diff_2([[0, 0, x, 0] for x in list(range(0,
6000))])),
       "GoogleBuy":  A1.predict(lag_diff_2([[0,  0,  0,  x]  for  x  in
list(range(0, 6000))])),
   }

   mobi_plot = {
       "PriceUA":  A2.predict(lag_diff_2([[x,  0,  0,  0]  for  x  in
list(range(0, 6000))])),
       "Nadavi":  A2.predict(lag_diff_2([[0,  x,  0,  0]  for  x  in
list(range(0, 6000))])),
       "XT": A2.predict(lag_diff_2([[0, 0, x, 0] for x in list(range(0,
6000))])),
       "GoogleBuy":  A2.predict(lag_diff_2([[0,  0,  0,  x]  for  x  in
list(range(0, 6000))])),
   }

   emag_plot = {
```

```
        "PriceUA":  A3.predict(lag_diff_2([[x,  0,  0,  0]  for  x  in
list(range(0, 6000))])),
        "Nadavi":  A3.predict(lag_diff_2([[0,  x,  0,  0]  for  x  in
list(range(0, 6000))])),
        "XT": A3.predict(lag_diff_2([[0, 0, x, 0] for x in list(range(0,
6000))])),
        "GoogleBuy":  A3.predict(lag_diff_2([[0,  0,  0,  x]  for  x  in
list(range(0, 6000))])),
    }
```

Лістинг 16. Проведення аналізу рекламних платяформ за допомою залежної моделі

```
    lex_plot_dep = {
        "PriceUA": B1.predict(ad_rows[0]),
        "Nadavi": B1.predict(ad_rows[1]),
        "XT": B1.predict(ad_rows[2]),
        "GoogleBuy": B1.predict(ad_rows[3]),
    }

    mobi_plot_dep = {
        "PriceUA": B2.predict(ad_rows[4]),
        "Nadavi": B2.predict(ad_rows[5]),
        "XT": B2.predict(ad_rows[6]),
        "GoogleBuy": B2.predict(ad_rows[7]),
    }

    emag_plot_dep = {
        "PriceUA": B3.predict(ad_rows[8]),
        "Nadavi": B3.predict(ad_rows[9]),
        "XT": B3.predict(ad_rows[10]),
        "GoogleBuy": B3.predict(ad_rows[11]),
    }
```

Окрім цих двух типів можливо побудувати «словники впливів», у яких будуть результати проведення лінійної регресії «зав’язаних» моделей. Це можливо через те що оцінка лінійної регресії – сума параметрів помножених на відповідні коефіцієнти моделі. Код реалізації цих словників виглядає наступним чином:

Лістинг 17. Словники впливів

```
lex_plot_m = {
    "PriceUA": B1.predict(ad_rows[4]),
    "Nadavi": B1.predict(ad_rows[5]),
    "XT": B1.predict(ad_rows[6]),
    "GoogleBuy": B1.predict(ad_rows[7]),
}

lex_plot_e = {
    "PriceUA": B1.predict(ad_rows[8]),
    "Nadavi": B1.predict(ad_rows[9]),
    "XT": B1.predict(ad_rows[10]),
    "GoogleBuy": B1.predict(ad_rows[11]),
}

mobi_plot_l = {
    "PriceUA": B2.predict(ad_rows[0]),
    "Nadavi": B2.predict(ad_rows[1]),
    "XT": B2.predict(ad_rows[2]),
    "GoogleBuy": B2.predict(ad_rows[3]),
}

mobi_plot_e = {
    "PriceUA": B2.predict(ad_rows[8]),
    "Nadavi": B2.predict(ad_rows[9]),
    "XT": B2.predict(ad_rows[10]),
    "GoogleBuy": B2.predict(ad_rows[11]),
}

emag_plot_l = {
    "PriceUA": B3.predict(ad_rows[0]),
    "Nadavi": B3.predict(ad_rows[1]),
    "XT": B3.predict(ad_rows[2]),
   "GoogleBuy": B3.predict(ad_rows[3]),
}

emag_plot_m = {
    "PriceUA": B3.predict(ad_rows[4]),
    "Nadavi": B3.predict(ad_rows[5]),
    "XT": B3.predict(ad_rows[6]),
```

```
    "GoogleBuy": B3.predict(ad_rows[7]),
}
```

Зберемо словники, які відносяться до кожного типу у відповідні їм словники:

Лістинг 18. Об’єднання словників з результатами для відображення

```
independent_plots = {
    "Mag1 ": lex_plot,
    "Mag2": mobi_plot,
    "Mag": emag_plot
}

dependent_plots = {
    "Mag1": lex_plot_dep,
    "Mag2": mobi_plot_dep,
    "Mag3": emag_plot_dep
}

influence_plots = {
    "Lexore --> Mobitek": lex_plot_m,
    "Lexore --> Emagaz": lex_plot_e,
    "Mobitek --> Lexore": mobi_plot_l,
    "Mobitek --> Emagaz": mobi_plot_e,
    "Emagaz --> Lexore": emag_plot_l,
    "Emagaz --> Mobitek": emag_plot_m,
}
```

Провівши лінійну регресію маємо результати схожі на наступний графік:

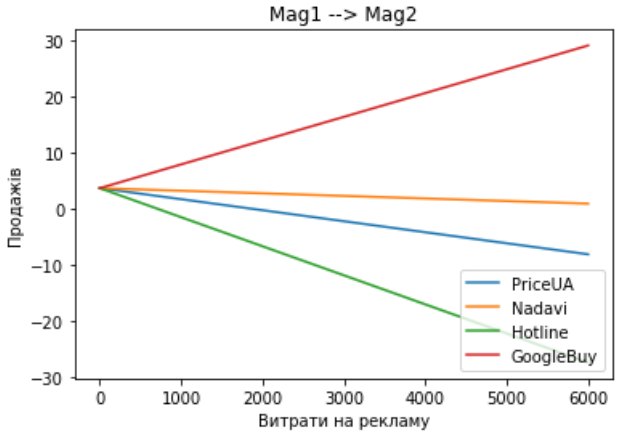


Рис 9. Вплив Mag1 на Mag2

На цьому графіку ми бачимо, скільки продажів отримує Mag2 в залежності від витрат на рекламу Mag1.

Окрім цього, при виконанні даної оцінки, відображається інформація, про результати:

Лістинг 19. Оцінка коефіцієнтів кількості продажів в залежності від витрат на рекламні платформи

```
Коеф. ефективності:

PriceUA: -0.001963
Початкове значення: 3.759489
Nadavi: -0.000458
Початкове значення: 3.759489
Hotline: -0.005175
Початкове значення: 3.759489
GoogleBuy: 0.004236
Початкове значення: 3.759489

Продажів на 1000 грн.:

PriceUA: -1.962745
Nadavi: -0.458157
Hotline: -5.174611
GoogleBuy: 4.235569

Коеф відносно найбільшого:

PriceUA: -0.463396
Nadavi: -0.108169
Hotline: -1.221704
GoogleBuy: 1.000000
```

4. РЕАЛІЗАЦІЯ ПЗ З ГРАФІЧНИМ ІНТЕРФЕЙСОМ

Набори програмного інструментарію, зазначені вище, дозволяють створити програму, яка дозволяє за допомогою програмного інтерфейсу змінювати вхідні параметри моделей та отримувати результат у реальному часі.

Для створення програмного забезпечення з графічним інтерфейсом, необхідно обрати відповідні програмні інструменти. Для виконання даного дипломного проекту була обрана бібліотека python pyqt5, яка дозволяє використовувати елементи фреймворку Qt для побудови інтерфейсу програми.

Найшвидший шлях побудови інтерфейсу включає в себе розмітку інтерфейсу у додатку Qt Designer, далі вихідний файл типу “.ui” повинен пройти процес трансляції у файл мови python.

Приклад команди, яка виконує трансляцію: «pyuic5 sales1.ui -o sales_ui.py»

Утиліта «pyuic5» поставляється разом з бібліотекою pyqt5. Після створення результату трансляції, необхідно створити файл з основною логікою графічної частини. У проекті даної роботи він названий «sales.py», також був створений файл, який містить основну логіку моделей даного ПЗ – «sales_model.py». Файли sales_ui.py і sales_model.py включаються у sales.py.

Програма має наступний вигляд:

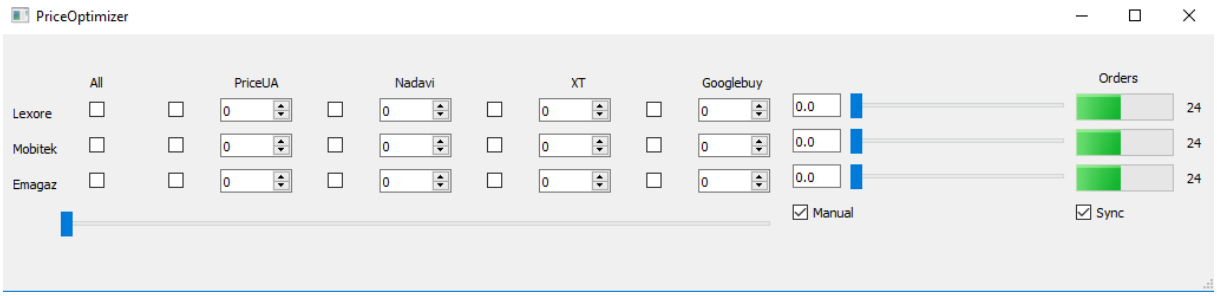


Рис 10. Графічний інтерфейс програми

Кількість продажів відображається справа, у кожній відповідній до назви зліва шкалі прогресу. Кожен «бігунок» лівіше шкали – рівень націнки

на товари з відповідним значенням у полі лівіше від нього (це поле можливо редагувати). Посередині інтерфейсу – значення витрат на рекламу, де у верхній частині відповідна назва рекламної платформи, а зліва – назва інтернет-магазину.

Якщо булеве поле Manual увімкнене:

Якщо відповідні булеві поля увімкнені, це означає що користування відповідною рекламною платформою відбувається і числове поле (яке може бути відредагованим як стрілками справа від значення, так і бігунцем знизу) включається у модель у незмінному вигляді, якщо булеве поле має виключене значення, на вхід у модель надходить значення 0.

Якщо булеве поле Manual вимкнене:

Якщо стан користування відповідними рекламними платформами увімкнений, на вхід у модель поступають значення витрат, які були оцінені іншою моделлю де на вхід іде націнка на товари, а на виході – кількість витрат. Якщо стан вимкнений, поступає нуль.

Булеве значення Sync увімкнене: очищення некоректних даних відбувається таким чином, що записи синхронізовані у часі (деякі дані втрачаються)

Булеве значення Sync вимкнене: очищення некоректних даних проходить не враховуючи синхронність (додаткової втрати даних не відбувається)

ВИСНОВКИ

При виконанні даного дипломного проекту, було створено програмну базу для реалізації оцінки економічних та математичних моделей за допомогою статистичних методів. Метод маржинальних флуктацій не показав достатньо точний результат, проте було успішно створене ПЗ з графічним інтерфейсом і реалізовані достатньо точні оцінки кількості продаж використовуючи об’єм витрат на рекламу. Даний програмний інструментарій, а також ПЗ з графічним інтерфейсом можливо вдосконалити провівши дослідження шляхів вдосконалення методу маржинальних флуктуацій. Також більшої точності можливо досягнути використавши якісно і кількісно інші дані, які будуть мати мінімальну кількість некоректних даних та охоплювати мінімум 1-2 роки.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1) Верхоглядова Н. І. Основи ціноутворення : навч. посіб. / Н. І. Верхоглядова, С. Б. Ільїна, Н. А. Іваннікова, Я. Я. Слабко [та ін.]. – К. : Кондор, 2007. – 252 с.
2) Дайновська С. М. Ціноутворення / С. М. Дайновська. – К. : ЦУЛ, 2009. – 320 с.
3) Корінєв В. Л. Маркетингова цінова політика : навч. посіб. / В. Л. Корінєв, М. Х. Корецький, О. І. Дацій. – К. : Центр учбової літератури, 2007. – 200 с.
4) Пінішко В. С. Ціно- і тарифоутворення : навч. посіб. / В. С. Пінішко. – Л. : Магнолія 2006, 2009. – 303 с.
5) Шкварчук Л. О. Ціноутворення : підручник / Л. О. Шкварчук. – К. : Кондор, 2008. – 460 с.
6) Python – documentation reference [Електронний ресурс] – Режим доступу: https://docs.python.org/3/library/index.html
7) Sklearn - Ordinary Least Squares [Електронний ресурс] – Режим доступу: https://scikit-learn.org/stable/modules/linear_model.html#ordinary-least-squares
8) Data science – Wikipedia [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Data_science
9) Ordinary least squares - Wikipedia [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Ordinary_least_squares
10) Matplotlib - pyplot.plot [Електронний ресурс] – Режим доступу: https://matplotlib.org/3.1.0/api/_as_gen/matplotlib.pyplot.plot.html
11) Machine learning – Wikipedia [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Machine_learning
12) Linear regression - Wikipedia [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Linear_regression
13) PyQt5 tutorial [Електронний ресурс] – Режим доступу: https://build-system.fman.io/pyqt5-tutorial
14) Pandas overview [Електронний ресурс] – Режим доступу: https://pandas.pydata.org/pandas-docs/stable/

15) A Simple Example of Pipeline in Machine Learning with Scikit-learn – Towards Data Science [Електронний ресурс] – Режим доступу: https://towardsdatascience.com/a-simple-example-of-pipeline-in-machine-learning-with-scikit-learn-e726ffbb6976

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"
Науковий керівник кафедри
_____ І.А. Дичка
"___" _____ 2018 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ АНАЛІЗУ ЕФЕКТИВНОСТІ ІНТЕРНЕТ-МАГАЗИНІВ НА ОСНОВІ СТАТИСТИЧНОЇ ОБРОБКИ ДАНИХ

Програма та методика тестування

ДП.045440-04-51

"ПОГОДЖЕНО"

Керівник проекту:

_____ В.І. Сущук-Слюсаренко

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____Єременко В.Є.

2018

ЗМІСТ

1. ОБ’ЄКТ ВИПРОБУВАНЬ

Програмне забезпечення з графічним інтерфейсом для моделювання кількості продажів інтернет-магазину при зміні його параметрів

2. МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

1) Можливість зміни булевих полів;

2) можливість зміни витрат на рекламні платформи за допомогою вводу числа у поле;

3) можливість зміни витрат на рекламні платформи за допомогою натискання на відповідні кнопки (стрілки);

4) можливість зміни витрат на рекламні платформи за допомогою пересування повзу;

5) можливість зміни маржі;

6) відображення результатів після зміни параметрів.

3. МЕТОДИ ТЕСТУВАННЯ

Тестування виконується методом Gray Box Testing. Перевіряється як код, так і безпосередньо програмний продукт на відповідність функціональним вимогам. Тестування відбувається на рівні «системного тестування».

Використовуються наступні методи:

1) функціональне тестування, зокрема на рівні Critical path test (базове тестування);
2) тестування інтерфейсу.

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Працездатність програмного забезпечення перевіряється наступним шляхом:

1) динамічного ручного тестування – введенням граничних та недопустимих значень в поля, які можна редагувати;
2) динамічного ручного тестування на відповідність функціональним вимогам;
3) статичного тестування коду;
4) тестування зручності використання;
5) тестування інтерфейсу.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2019 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ АНАЛІЗУ ЕФЕКТИВНОСТІ ІНТЕРНЕТ-МАГАЗИНІВ НА ОСНОВІ СТАТИСТИЧНОЇ ОБРОБКИ ДАНИХ

Керівництво користувача

ДП.045440-05-34

"ПОГОДЖЕНО"

Керівник проекту:

_____ В.І Сущук-Слюсаренко

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____Єременко В.Є.

2019

ЗМІСТ

1. Вимоги для використання ПЗ з графічним інтерфейсом

Розроблене при написанні дипломної роботи програмне забезпечення має наступні обов’язкові вимоги:

- Python 3.6 і вище
- Pip актуальної версії
- Pandas
- Numpy
- Sklearn
- Matpotlib
- Pyqt5
- ОС Windows

2. Керівництво використання ПЗ з графічним інтерфейсом

Якщо обов’язкові вимоги дотримані, можливий запуск програмного забезпечення за допомогою файлу «sales.bat».

Графічний інтерфейс має наступний вигляд:

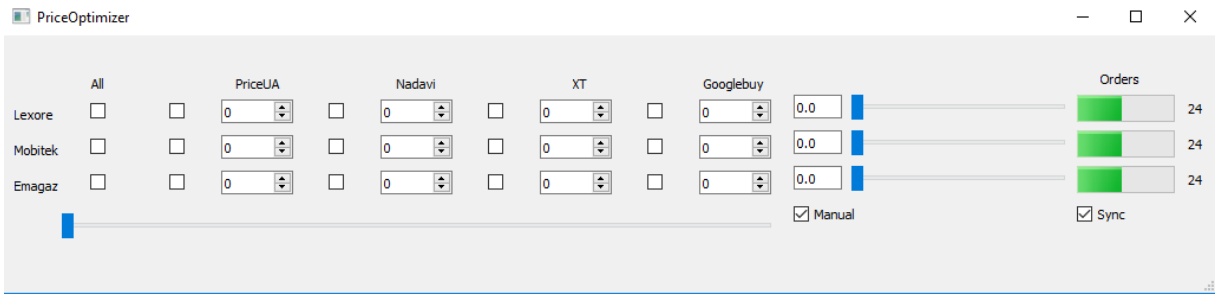


Рис. 1. Графічний інтерфейс програми

Кількість продажів відображається справа, у кожній відповідній до назви зліва шкалі прогресу. Кожен «бігунок» лівіше шкали – рівень націнки на товари з відповідним значенням у полі лівіше від нього (це поле можливо редагувати). Посередині інтерфейсу – значення витрат на рекламу, де у верхній частині відповідна назва рекламної платформи, а зліва – назва інтернет-магазину.

Якщо булеве поле Manual увімкнене:

- Якщо відповідні булеві поля увімкнені, це означає що користування відповідною рекламною платформою відбувається і числове поле (яке може бути відредагованим як стрілками справа від значення, так і бігунцем знизу) включається у модель у незмінному вигляді, якщо булеве поле має виключене значення, на вхід у модель надходить значення 0.

Якщо булеве поле Manual вимкнене:

- Якщо стан користування відповідними рекламними платформами увімкнений, на вхід у модель поступають значення витрат, які були оцінені іншою моделлю де на вхід іде націнка на товари, а на виході – кількість витрат.
- Якщо стан вимкнений, поступає нуль.

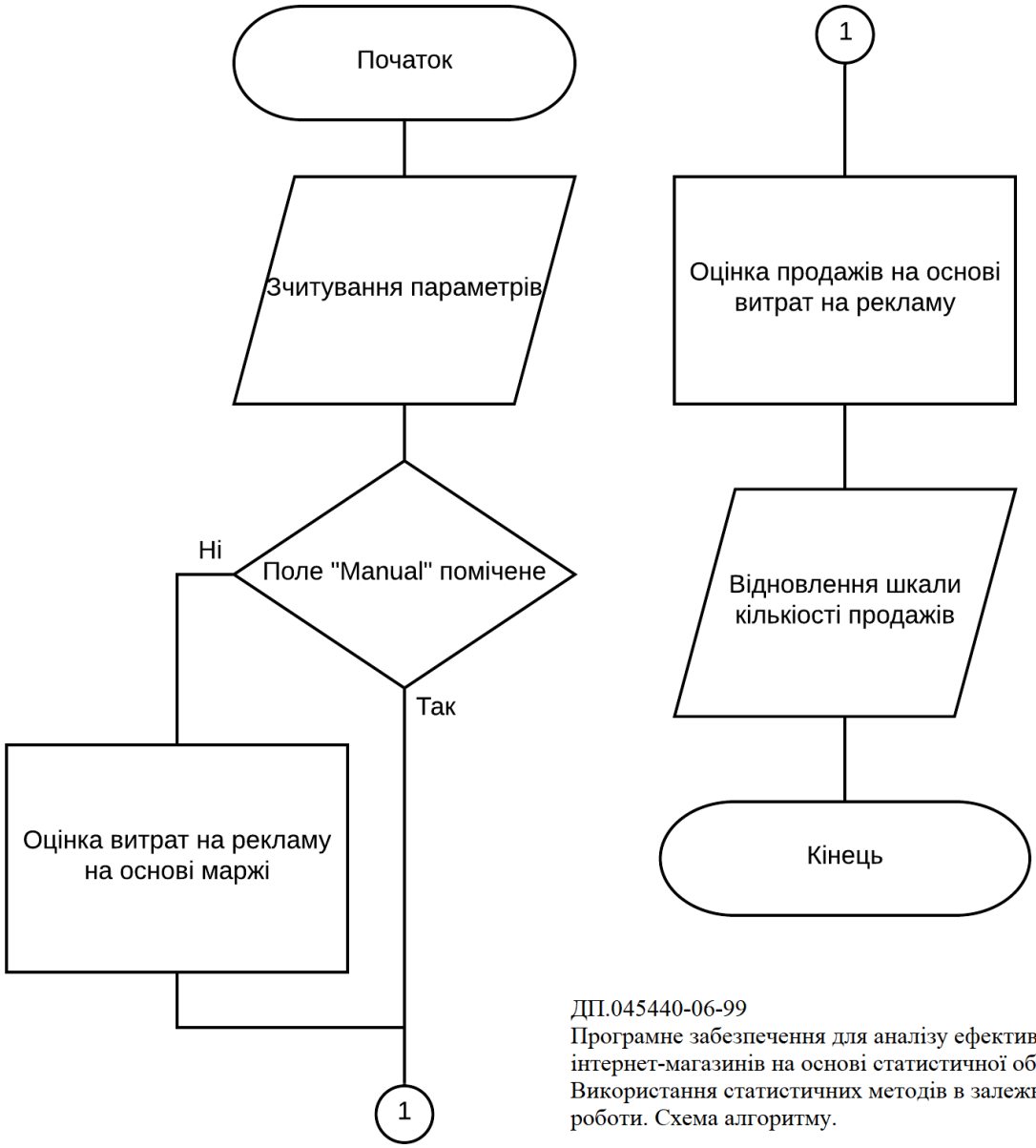
Булеве значення Sync увімкнене: очищення некоректних даних відбувається таким чином, що записи синхронізовані у часі (деякі дані втрачаються)

Булеве значення Sync вимкнене: очищення некоректних даних проходить не враховуючи синхронність (додаткової втрати даних не відбувається)

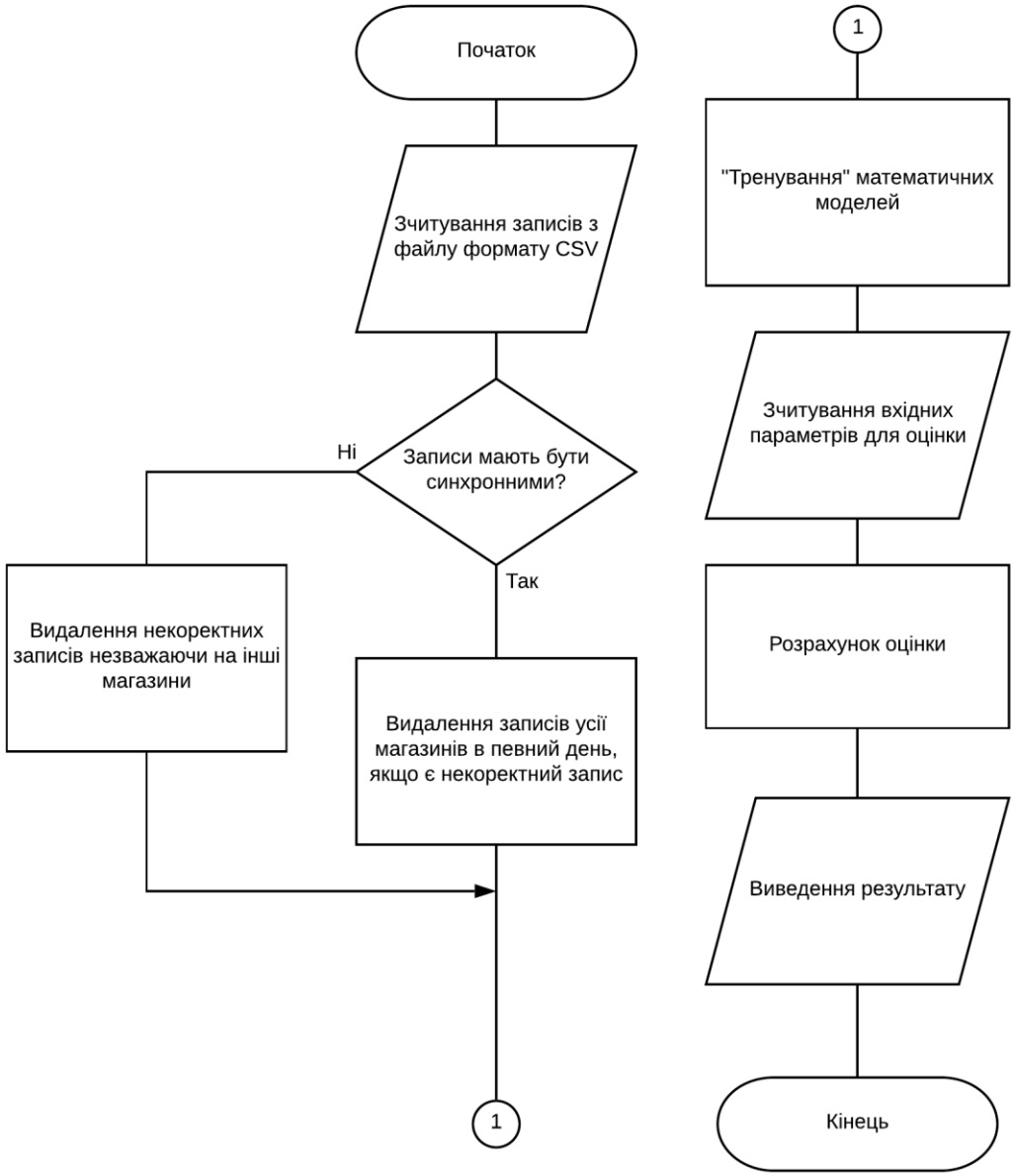
ДОДАТКИ

Додаток 1

Копії графічних матеріалів



ДП.045440-06-99
Програмне забезпечення для аналізу ефективності роботи інтернет-магазинів на основі статистичної обробки даних. Використання статистичних методів в залежності від режиму роботи. Схема алгоритму.



ДП.045440-07-99
Програмне забезпечення для аналізу ефективності Інтернет-магазинів на основі статистичної обробки даних.
Обробка даних та проведення оцінки.
Схема алгоритму.

Оцінка роботи статистичних методів

Предсказание заказов			
Магазины отдельно	Lexore	Mobitek	Emagaz
сум. рек	6,54	12,32	14,92
рек. отдельно	6,78	9,2	16,55
сум. рек и маржа	7,44	10,56	15,92
активная рек + маржа	7,73	8,13	11,49
вся рек + маржа	8,22	8,53	11,45
Оптимально	**6,54**	**8,13**	**11,45**
Средняя точность	**64,00%**		

Магазины завязаны	Lexore	Mobitek	Emagaz
сум. рек	7,03	9,9	15,99
рек. отдельно	13,58	9,24	15,67
сум. рек и маржа	8,12	6,98	16,01
активная рек + маржа	14,51	6,9	11,11
вся рек + маржа	15,6	8,82	12,18
Оптимально	**7,03**	**6,9**	**11,11**
Средняя точность	**71,30%**		

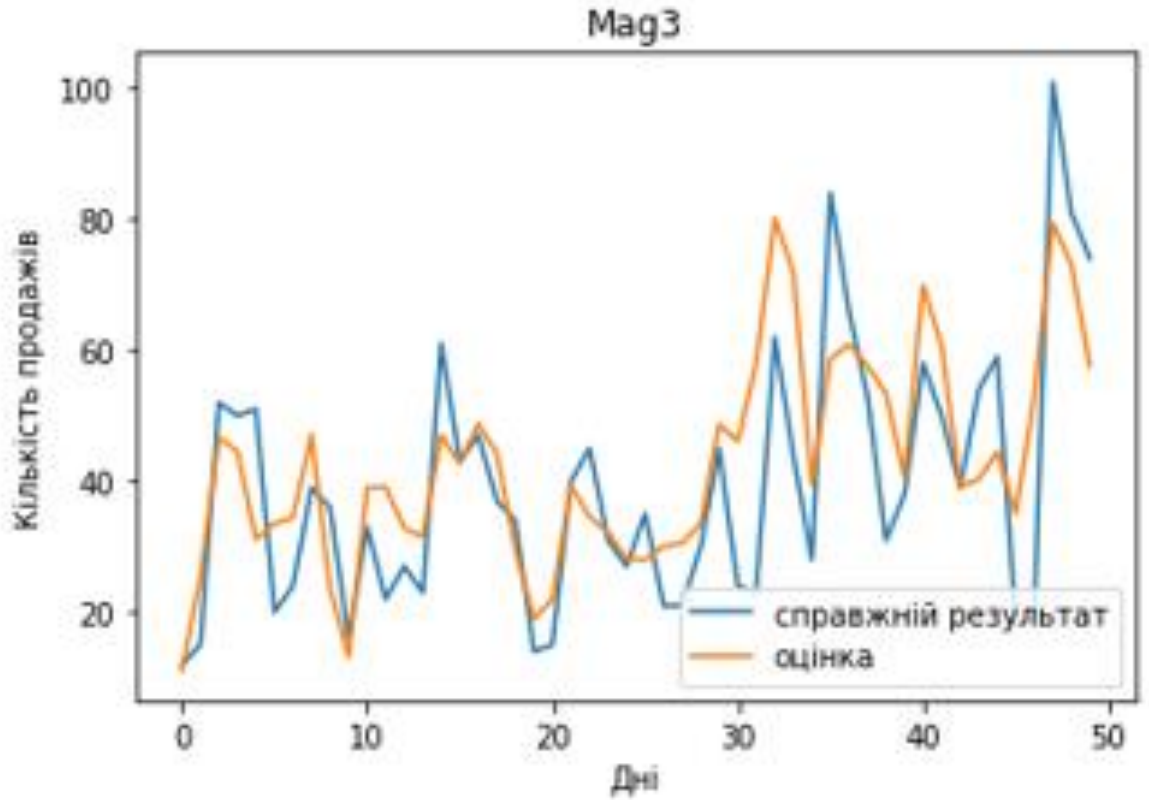
Оптимально общ.	**71%**	**70%**	**75%**
Итого	**72%**		

Предсказание просмотров			
Магазины отдельно	Lexore	Mobitek	Emagaz
	795	929	736
	54%	-61%	60%

Магазины завязаны	Lexore	Mobitek	Emagaz
	705	926	750
	59%	-61%	61%

Єременко В.Є. група КП-52

Ілюстрація роботи статистичних методів



Єременко В.Є., група КП-52

Додаток 2

Лістинг файлу sales.py з логікою графічного інтерфейсу

```
from sales_model import *

class MyWin(QtWidgets.QMainWindow):
    def __init__(self, parent=None):
        QtWidgets.QWidget.__init__(self, parent)
        self.lg, self.means, self.lg_a, self.vals_decomposed = train()
        self.poly = PolynomialFeatures(degree=2)
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)
        self.selectedSB = None
        self.cbs = {
            "Lexore": [
                self.ui.cb_l_priceua,
                self.ui.cb_l_nadavi,
                self.ui.cb_l_xt,
                self.ui.cb_l_googlebuy
            ],
            "Mobitek": [
                self.ui.cb_m_priceua,
                self.ui.cb_m_nadavi,
                self.ui.cb_m_xt,
                self.ui.cb_m_googlebuy
            ],
            "Emagaz": [
                self.ui.cb_e_priceua,
                self.ui.cb_e_nadavi,
                self.ui.cb_e_xt,
                self.ui.cb_e_googlebuy
            ]
        }

        self.sbs = {
            "Lexore": [
                self.ui.sb_l_p,
                self.ui.sb_l_n,
                self.ui.sb_l_xt,
                self.ui.sb_l_g
            ],
            "Mobitek": [
                self.ui.sb_m_p,
                self.ui.sb_m_n,
                self.ui.sb_m_xt,
                self.ui.sb_m_g
            ],
            "Emagaz": [
                self.ui.sb_e_p,
                self.ui.sb_e_n,
                self.ui.sb_e_xt,
                self.ui.sb_e_g
            ]
        }
```

```
            for v in self.sbs:
                for i in range(0, 3):

self.sbs[v][i].setMaximum(max(self.vals_decomposed[v][i+1]))
                    minval = min(self.vals_decomposed[v][i+1])
                    self.sbs[v][i].setMinimum(minval)
                    self.sbs[v][i].setValue(minval)

            self.vals = {
                "Lexore": [0, 0, 0, 0],
                "Mobitek": [0, 0, 0, 0],
                "Emagaz": [0, 0, 0 ,0]
            }
            # ????? ??????????? ??????? ??????? ?? ??????
            self.ui.sb_l_p.valueChanged.connect(self.updateOrders)
            self.ui.sb_l_n.valueChanged.connect(self.updateOrders)
            self.ui.sb_l_xt.valueChanged.connect(self.updateOrders)
            self.ui.sb_m_g.valueChanged.connect(self.updateOrders)
            self.ui.sb_m_p.valueChanged.connect(self.updateOrders)
            self.ui.sb_m_n.valueChanged.connect(self.updateOrders)
            self.ui.sb_m_xt.valueChanged.connect(self.updateOrders)
            self.ui.sb_e_g.valueChanged.connect(self.updateOrders)
            self.ui.sb_e_p.valueChanged.connect(self.updateOrders)
            self.ui.sb_e_n.valueChanged.connect(self.updateOrders)
            self.ui.sb_e_xt.valueChanged.connect(self.updateOrders)

self.ui.hslid_selected_ad.valueChanged.connect(self.updateSBbySlider)
            self.ui.cb_manual.stateChanged.connect(self.updateOrders)
            self.ui.cb_sync.stateChanged.connect(self.retrain)
            self.ui.cb_l_xt.stateChanged.connect(self.updateOrders)
            self.ui.cb_l_nadavi.stateChanged.connect(self.updateOrders)
            self.ui.cb_l_priceua.stateChanged.connect(self.updateOrders)

self.ui.cb_l_googlebuy.stateChanged.connect(self.updateOrders)
            self.ui.cb_m_xt.stateChanged.connect(self.updateOrders)
            self.ui.cb_m_nadavi.stateChanged.connect(self.updateOrders)
            self.ui.cb_m_priceua.stateChanged.connect(self.updateOrders)

self.ui.cb_m_googlebuy.stateChanged.connect(self.updateOrders)
            self.ui.cb_e_xt.stateChanged.connect(self.updateOrders)
            self.ui.cb_e_nadavi.stateChanged.connect(self.updateOrders)
            self.ui.cb_e_priceua.stateChanged.connect(self.updateOrders)

self.ui.cb_e_googlebuy.stateChanged.connect(self.updateOrders)
            self.ui.cb_l_all.stateChanged.connect(self.LexoreAll)
            self.ui.cb_m_all.stateChanged.connect(self.MobitekAll)
            self.ui.cb_e_all.stateChanged.connect(self.EmagazAll)

self.ui.hslid_mar_lex.valueChanged.connect(self.LexUpdBySlider)

self.ui.hslid_mar_mob.valueChanged.connect(self.MobitekUpdBySlider)

self.ui.hslid_mar_emag.valueChanged.connect(self.EmagazUpdBySlider)
```

```
self.ui.ledit_mar_lex.editingFinished.connect(self.updateOrders)

self.ui.ledit_mar_mob.editingFinished.connect(self.updateOrders)

self.ui.ledit_mar_emag.editingFinished.connect(self.updateOrders)

       # ???? ?????? ??????? ??????? ???????????
       # ??? ??????? ?? ??????

       def focusInEvent(self, *args, **kwargs):
           self.updateOrders()

       def getVals(self):
           for i in range(0, 4):
               if self.cbs["Lexore"][i].isChecked():
                   self.vals["Lexore"][i]                                  =
self.sbs["Lexore"][i].value()
               if self.cbs["Mobitek"][i].isChecked():
                   self.vals["Mobitek"][i]                                 =
self.sbs["Mobitek"][i].value()
               if self.cbs["Emagaz"][i].isChecked():
                self.vals["Emagaz"][i] = self.sbs["Emagaz"][i].value()

       def setVals(self, vals):
           for i in range(0, 4):
               if self.cbs["Lexore"][i].isChecked():

self.sbs["Lexore"][i].setValue(self.vals["Lexore"][i])
               else:
                   self.sbs["Lexore"][i].setValue(0)
               if self.cbs["Mobitek"][i].isChecked():

self.sbs["Mobitek"][i].setValue(self.vals["Mobitek"][i])
               else:
                   self.sbs["Mobitek"][i].setValue(0)
               if self.cbs["Emagaz"][i].isChecked():

self.sbs["Emagaz"][i].setValue(self.vals["Emagaz"][i])
               else:
                   self.sbs["Emagaz"][i].setValue(0)

       def updateSBbySlider(self):
           if self.selectedSB is not None:

self.selectedSB.setValue(self.ui.hslid_selected_ad.value())

       def updateOrders(self):
           selected = self.focusWidget()
           if type(selected) == type(self.ui.sb_l_p):
               found = False
               for v in self.sbs:
                   print(v)
                   for i in range(0, 4):
```

```
                        if         self.sbs[v][i].objectName()         ==
selected.objectName():
                            self.selectedSB = selected

self.ui.hslid_selected_ad.setValue(self.vals[v][i])

self.ui.hslid_selected_ad.setMaximum(selected.maximum())
                            found = True
                            break
                    if found:
                        break

            if self.ui.cb_manual.isChecked():
                self.getVals()

            lex_mar = float(self.ui.ledit_mar_lex.text())
            mobi_mar = float(self.ui.ledit_mar_mob.text())
            emag_mar = float(self.ui.ledit_mar_emag.text())
            p_lex = self.lg_a["lex"]
            p_mobi = self.lg_a["mobi"]
            p_emag = self.lg_a["emag"]
            # print(lex_mar)
            # try:
            #     pepix = p_lex[0].predict(lex_mar)
            # except Exception as e:
            #     print(e)
            if self.ui.cb_manual.isChecked():
                lex_flags = self.vals["Lexore"]
                mobi_flags = self.vals["Mobitek"]
                emag_flags = self.vals["Emagaz"]
            else:
                lex_flags        =        [p_lex[0].predict([[lex_mar]]),
p_lex[1].predict([[lex_mar]]), p_lex[2].predict([[lex_mar]]),
                             p_lex[3].predict([[lex_mar]])]
                mobi_flags       =       [p_mobi[0].predict([[mobi_mar]]),
p_mobi[1].predict([[mobi_mar]]),
                              p_mobi[2].predict([[mobi_mar]]),
                              p_mobi[3].predict([[mobi_mar]])]
                emag_flags       =       [p_emag[0].predict([[emag_mar]]),
p_emag[1].predict([[emag_mar]]),
                              p_emag[2].predict([[emag_mar]]),
                              p_emag[3].predict([[emag_mar]])]

            vals = {
                "Lexore": lex_flags,
                "Mobitek": mobi_flags,
                "Emagaz": emag_flags
            }

            self.setVals(vals)

            lex_flags = [a*b for a, b in zip(lex_flags, [int(x.isChecked())
for x in self.cbs["Lexore"]])]
```

```
            mobi_flags = [a * b for a, b in zip(mobi_flags,
[int(x.isChecked()) for x in self.cbs["Mobitek"]])]
            emag_flags = [a * b for a, b in zip(emag_flags,
[int(x.isChecked()) for x in self.cbs["Emagaz"]])]
            print([int(x.isChecked()) for x in self.cbs["Emagaz"]])
            lex_x = lex_flags + [lex_mar]

            mobi_x = mobi_flags + [mobi_mar]

            emag_x = emag_flags + [emag_mar]
            # lex_x = [a*b for a, b in zip(self.means["lex"].copy(),
[int(x.isChecked()) for x in self.cbs["Lexore"]])] +
[float(self.ui.ledit_mar_mob.text())]
            # mobi_x = [a*b for a, b in zip(self.means["mobi"].copy(),
[int(x.isChecked()) for x in self.cbs["Mobitek"]])] +
[float(self.ui.ledit_mar_mob.text())]
            # emag_x = [a*b for a, b in zip(self.means["emag"].copy(),
[int(x.isChecked()) for x in self.cbs["Emagaz"]])] +
[float(self.ui.ledit_mar_emag.text())]
            T1, T2, T3 = method_transform(7, [lex_x], [mobi_x], [emag_x])

            X = T1[0] + T2[0] + T3[0]
            X = [[float(x) for x in X]]
            print(X)
            # print("THIS IS X!!")
            R1 = self.lg[0].predict(X)
            R2 = self.lg[1].predict(X)
            R3 = self.lg[2].predict(X)
            print("R1: {:f} R2: {:f} R3: {:f}".format(R1[0], R2[0], R3[0]))
            self.ui.pBar_o_lex.setValue(R1)
            self.ui.pBar_o_mon.setValue(R2)
            self.ui.pBar_o_emag.setValue(R3)

        def retrain(self):
            self.lg, self.means, self.lg_a, self.vals_decomposed =
train(self.ui.cb_sync.isChecked())
            self.updateOrders()
        def LexoreAll(self):
            if self.ui.cb_l_all.isChecked():
                    for cb in self.cbs["Lexore"]:
                        cb.setChecked(True)
            else:
                for cb in self.cbs["Lexore"]:
                    cb.setChecked(False)
        def MobitekAll(self):
            if self.ui.cb_m_all.isChecked():
                    for cb in self.cbs["Mobitek"]:
                        cb.setChecked(True)
            else:
                for cb in self.cbs["Mobitek"]:
                    cb.setChecked(False)
        def EmagazAll(self):
            if self.ui.cb_e_all.isChecked():
                    for cb in self.cbs["Emagaz"]:
```

```
                        cb.setChecked(True)
            else:
                for cb in self.cbs["Emagaz"]:
                    cb.setChecked(False)
        def LexUpdBySlider(self):

self.ui.ledit_mar_lex.setText(str(self.ui.hslid_mar_lex.value()/10))
            self.updateOrders()
        def MobitekUpdBySlider(self):

self.ui.ledit_mar_mob.setText(str(self.ui.hslid_mar_mob.value()/10))
            self.updateOrders()
        def EmagazUpdBySlider(self):

self.ui.ledit_mar_emag.setText(str(self.ui.hslid_mar_emag.value()/10))
            self.updateOrders()

    if __name__=="__main__":
        app = QtWidgets.QApplication(sys.argv)
```

Додаток 3

Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ АНАЛІЗУ ЕФЕКТИВНОСТІ ІНТЕРНЕТ-МАГАЗИНІВ НА ОСНОВІ СТАТИСТИЧНОЇ ОБРОБКИ ДАНИХ

Виконав: Єременко Владислав Євгенійович

Науковий керівник: ст. вик. Вікторія Ігорівна Сущук-Слюсаренко

Київ – 2019

АКТУАЛЬНІСТЬ

З настанням інформаційної епохи, автоматизація та використання цифрових технологій стали потужними двигунами комерційного успіху. Ці фактори вплинули й на сектор продажів.

На сьогоднішній день інтернет-комерція є дуже розвинутою і важливою галуззю, але питання аналітики ведення такого бізнесу залишається відкритим. Особливо це стосується малого та середнього підприємництва, які не можуть дозволити собі дорогі проекти націлені на використання великих об’ємів даних.

Таким чином, дана тема є вкрай актуальною.

ПОСТАНОВКА ЗАДАЧІ

Дослідити можливість використання статистичних методів та методу маржинальних флуктуацій для оцінки ефективності роботи інтернет-магазину. Розробити відповідне ПЗ з графічним інтерфейсом.

ЗАВДАННЯ

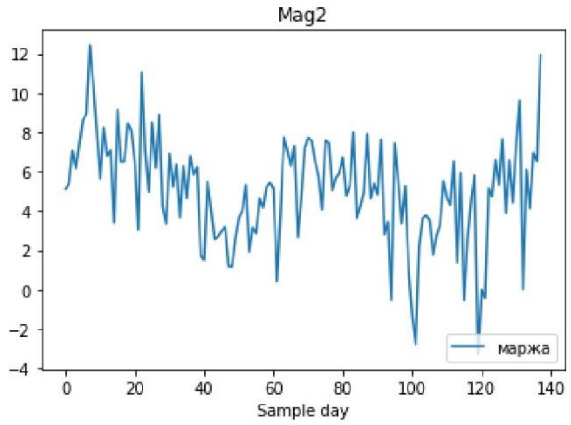
Завдання:

1. Дослідити можливість використання статистичних методів для оптимізації продажів інтернет-магазину
2. Проаналізувати ефективність методу маржинальних флуктуацій
3. Оцінити можливість оптимізації рекламних платформ
4. Створити ПЗ з графічною оболонкою

ДАНІ

Дані були отримані з аналітики рекламних платформ наведених вище. Націнка на товари є глобальною.
Кожен запис – звіт за день роботи, містить у собі витрати на платформи, націнку у відсотках та кількість продажів.
Період запису звітів – з травня по вересень 2018 року.
138 записів по 3 магазинам.
Тип товарів однаковий.
Якщо магазини користуються тими ж самими рекламними платформами – вони конкуренти.

МАРЖИНАЛЬНІ ФЛУКТУАЦІЇ



Періодичні випадкові зміни націнки. При зміні продажів відповідно до зміни націнки можливо отримати залежність кількості продажів від ціни.

ВИКОРИСТАНІ ТЕХНОЛОГІЇ

- Мова програмування – Python
- Модуль обробки даних – Pandas
- Модуль стат. методів – Sklearn
- Розробка ПЗ з граф. Інтерфейсом – PyQt5

ОБРАНА МЕТОДИКА

Основним обраним методом для роботи був обраний метод лінійної регресії:

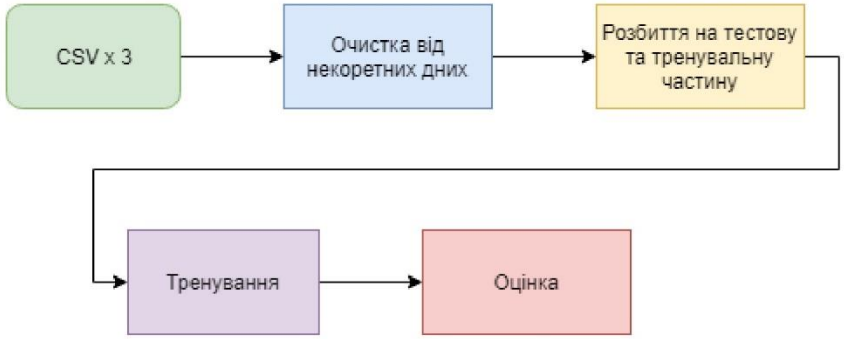
- Найкраща з малими обсягами даних
- Результат – лінійний многочлен, можливо дізнатися вплив кожного параметру на результат.
- Експерименти показали, що даний метод має найбільшу точність при проведенні виконанні роботи

СХЕМА РОБОТИ

Загальне число записів – N

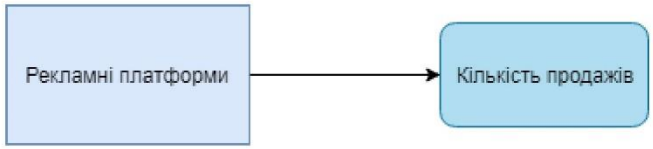
Число записів для «тренування» – (N-50)

Число записів для тестування – 50

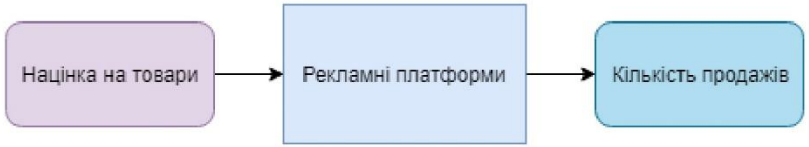


РОБОТА СТАТИСТИЧНИХ МЕТОДІВ

Прямий режим

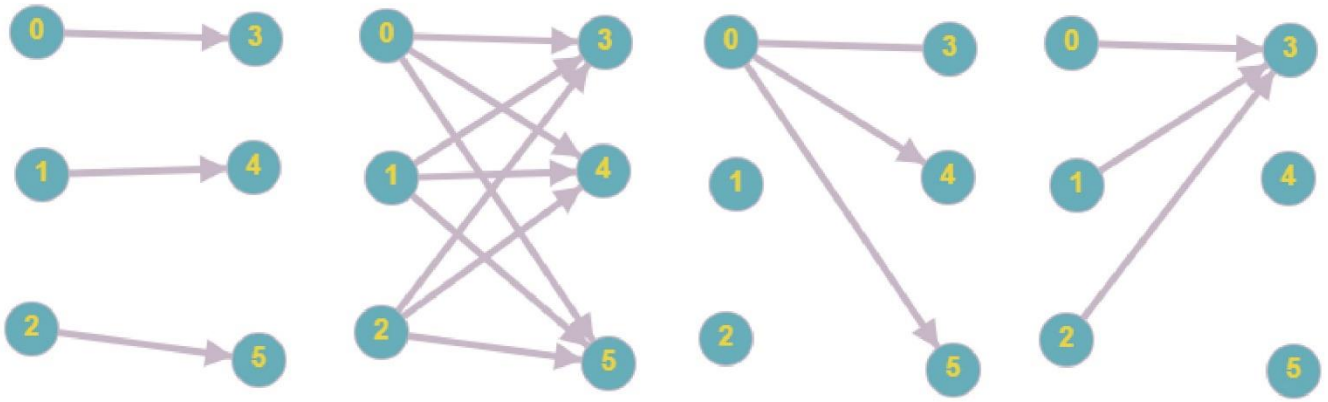


Режим подвійної оцінки



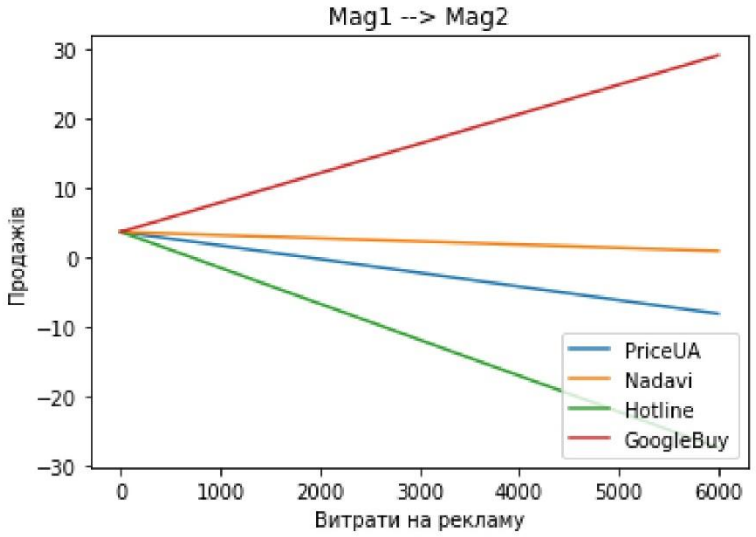
СПІЛЬНІ ПАРАМЕТРИ

При створенні математичних моделей для вирішення даної задачі можливі два підходи. Один з них – створити три моделі де кожна з них буде мати параметри відповідного магазину. Проте є й інший – об’єднати параметри усіх магазинів і використати цей набір для кожного.



ОЦІНКА КОНКУРЕНІЇ

Таким чином можливо отримати вплив одних магазинів на інші і оцінити конкуренцію між ними.



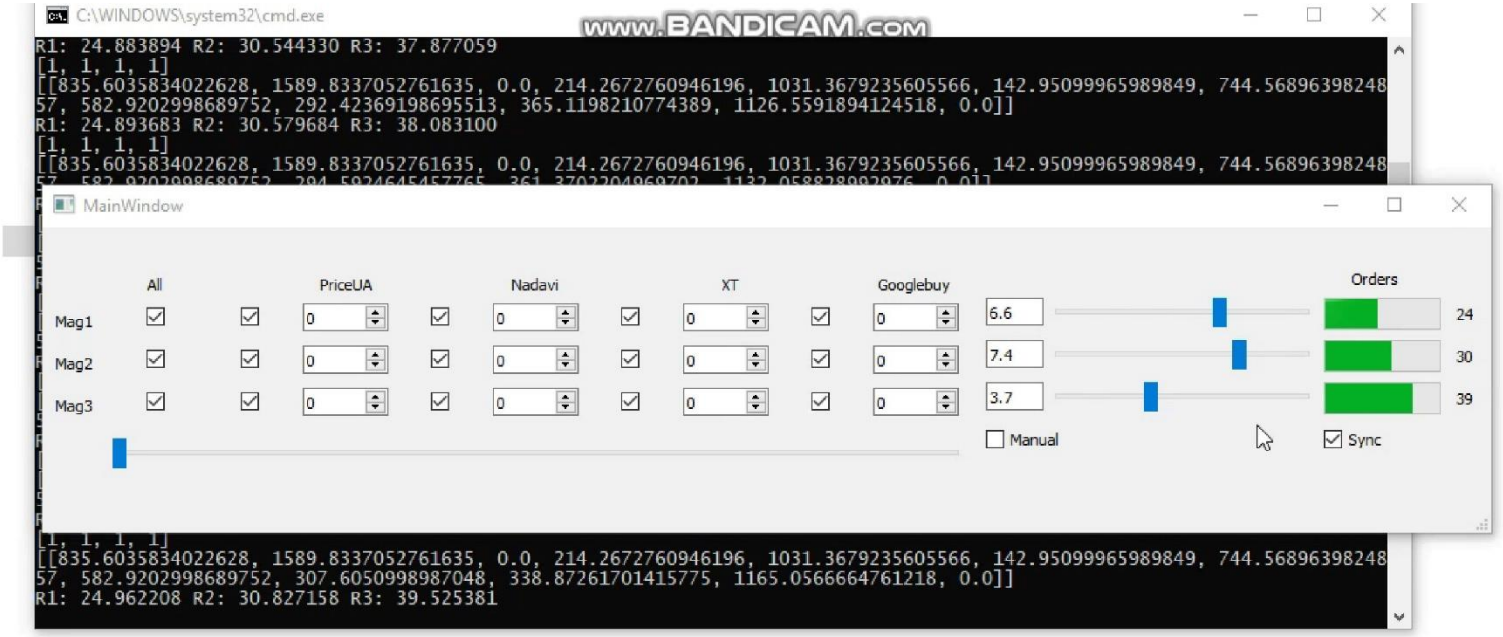
ОЦІНКА У РЕАЛЬНОМУ ЧАСІ

Використав повний набір даних як тренувальний, можливо отримати максимально точну математичну модель. Так як подібні статистичні методи потребують складних обчислень тільки при створенні самої моделі, можливо створити ПЗ, яке буде моделювати усю структуру у реальному часі.

Інструментарій для розробки ПЗ з графічним інтерфейсом Qt має великий функціонал для швидкої розробки.

Бібліотека PyQt5 дозволяє створювати файли з кодом Python, який буде запускати інтерфейс Qt. Логіку можливо дописати самостійно

ПЗ З ГРАФІЧНИМ ІНТЕРФЕЙСОМ



РЕЗУЛЬТАТИ

1) Статистичні методи показали задовільну точність, проте високою є залежність від об’єму даних.
2) Метод маржинальних флуктуацій опинився неефективним і його використання не рекомендується.
3) Оцінка ефективності використання рекламних платформ показала гарні результати.
4) Було розроблене ПЗ з графічним інтерфейсом на основі результатів дослідження, яке здатне моделювати вплив параметрів на результати роботи інтернет-магазину.

ВИСНОВКИ

Статистичні методи показали можливість оцінювати показники економічної моделі. При достатньому об’ємі та якості даних, розробка ПЗ, яке моделює економічний стан підприємства є доцільним рішенням.

Проте отримання якісних даних швидко доволі важко, навіть використовуючи такі імпровізовані способи, як створення маржинальних флуктуацій.

Результати цього проекта показали, що даний напрям досліджень і розробок має великий потенціал і може відкрити для підприємців гарний метод пошуку ефективних рішень у компаніях.

Дякую за увагу!